SEPTEMBER 1999

;login:

The USENIX Association Magazine

Special Issue on Intrusion Detection

SPECIAL ISSUE

Intrusion Detection

inside:

features:

USENIX The Advanced Computing Systems Association

USENIX *Upcoming* Events

2nd Conference on Domain-Specific Languages (DSL '99)

In cooperation with ACM SIGPLAN and SIGSOFT
October 3-5, 1999
Omni Hotel, Austin, Texas, USA
Web site: http://www.usenix.org/events/dsl99

2nd USENIX Symposium on Internet Technologies and Systems (USITS '99)

Co-sponsored by the IEEE Computer Society Technical Committee on the Internet
October 11-14, 1999
Regal Harvest House Hotel, Boulder, Colorado, USA
Web site: http://www.usenix.org/events/usits99

3rd Annual Atlanta Linux Showcase

Co-sponsored by USENIX, Atlanta Linux Enthusiasts, and Linux International
October 12-16, 1999
Cobb Galleria, Atlanta, Georgia, USA
Web site: http://www.linuxshowcase.org

13th Systems Administration Conference (LISA '99)

Sponsored by SAGE, The System Administrators Guild
November 7-12, 1999
Seattle Convention Center, Seattle, Washington, USA
Web site: http://www.usenix.org/events/lisa99

NordU 2000—2nd Nordic EurOpen/USENIX Conference

February 8-11, 2000
Malmö, Sweden
Web site: http://www.nordu.org/NordU2000/
Submissions due: September 13, 1999

7th USENIX Tcl/Tk Conference (Tcl/2k)

February 14-18, 2000
Austin, Texas, USA
Web site: http://www.usenix.org/events/tcl2k

SANS 2000—9th Annual System Administration, Networking, and Security Conference

Co-sponsored by the SANS Institute and SAGE
March 21-27, 2000
Orlando, Florida, USA
Web site: http://www.sans.org
Submissions due: September 15, 1999

SANE 2000—2nd International System Administration and Networking Conference

Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet
May 22-25, 2000
Maastricht, The Netherlands
Web site: http://www.nluug.nl/events/sane2000/
Submissions due: November 1, 1999

2000 USENIX Annual Technical Conference

June 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
Web site: http://www.usenix.org/events/usenix2000
Submissions due: November 29, 1999

LISA-NT--3rd Large Installation System Administration of Windows NT/2000 Conference

July 30-August 2, 2000
Seattle, Washington
Web site: http://www.usenix.org/events/lisa-nt2000
Submission due: February 16, 2000

4th USENIX Windows Operating Systems Symposium

August 3-4, 2000
Seattle, Washington
Web site: http://www.usenix.org/events/usenix-win2000

9th USENIX Security Symposium

August 14-17, 2000
Denver, Colorado, USA
Web site: http://www.usenix.org/events/sec2000
Submissions due: February 10, 2000

For a complete list of future USENIX events, access *http://www.usenix.org/events*

contents

Cover: The I-Map BOF at the Network Administration Conference

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,
the Association's magazine, published eight to ten times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to ;login: online
from October 1997 to last month.
<www.usenix.org/publications/login/login.html>

Access to papers
from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web.
<www.usenix.org/publications/library/index.html>.

The right to vote
on matters affecting the Association, its bylaws, election of its directors and officers.

Optional membership
in SAGE, the System Administrators Guild.

Discounts on registration fees
for all USENIX conferences.

Discounts
on the purchase of proceedings and CD-ROMS from USENIX conferences.

Savings
(see <http://usenix.org/membership/membership.html> for details)
10% off all Academic Press Professional books.
10% off BSDI, Inc. "personal" products.
10% off Morgan Kaufmann books.
20% off New Riders/Cisco Press/MTP books.
10% off OnWord Press publications.
10% off The Open Group publications.
20% off O'Reilly & Associates publications.
$10.00 off Prime Time Freeware publications and software.
10% off Wiley Computer Publishing books.

Special subscription rates
(see <http://usenix.org/membership/membership.html> for details)
$45 subscription to *IEEE Concurrency* (regularly $88).
15% off subscription to *The Linux Journal*.
20% off subscription to any Sage Science Press journals.

For more information regarding membership or benefits please contact
<office@usenix.org>
Phone: 510 528 8649

in this issue . . .

by Rik Farrow
Special Issue Editor
<rik@spirit.com>

This special edition of *;login:* focuses on network issues, particularly intrusion-detection systems (IDSes). The summaries from the First Conference on Network Administration, as well as from the Workshop on Intrusion Detection and Network Monitoring, will perhaps whet your appetite for what follows them.

Steve Romig, a security specialist at The Ohio State University, shares some of the work he and others have been doing with flow accounting. Some Cisco routers use speeded-up mechanisms for routing and access-control lists. By collecting data from this activity, you can, to a large degree, summarize network traffic. With any large network, this is somewhat like drinking from a firehose, so Steve also talks about some of the data-summary tools they have developed to make sense of the logs.

David Brumley, a specialist for the Stanford University Network Security Team, shares some of his experience in cleaning up after incidents by describing rootkits. Rootkits have been around in some form for more than five years, but David has seen some new twists that I certainly hadn't heard of before. Rootkits help to make intruders invisible, but knowing what the kits do and how they work can help you discover these hidden packages.

Paul Brutch, a graduate student at Texas A&M and a USENIX scholar, writes about his experience in a hacking class. Unlike the classes that used to be given in Dutch universities, the purpose of the class was not to break into other organizations' computers, but to penetrate a lab network before the defending group of sysadmins could secure the systems. Guess who won? His story of techniques and tactics will help you understand what you are up against (if you plan on not being hacked).

Mudge, a hacker (in the older sense, and perhaps the newer as well), is best known for revealing various security vulnerabilities in Microsoft protocols. But I was intrigued when I learned that Mudge and some others at L0pht had been writing N-Code modules for Network Flight Recorder, so I asked Mudge if he would mind writing about what had so inspired him. Mudge shares his perspective on the state of ID systems – a perspective that is not far from the laments of Vern Paxson and Ed Amoroso in their invited talks during the ID workshop. He also reveals his plans for a next step in ID software.

Tim Bass (a consultant) and Dave Gruber (an Air Force communications squadron commander) discuss their own, forward spin on ID systems. Their article uses several analogies to real-world systems, but the point I came away with is that ID systems will never really work in real-world environments. Tim, of course, will happily argue this point with you, and perhaps, some day, he will be right.

John Sellens completes his series on Reliability with an article about security. Sellens' main point is that you cannot have reliable systems unless they are secure. I'd go further, and say that your systems are not well administered unless they are secure, but that's just my opinion. Sellens provides lots of good advice, including some ideas that you just might have overlooked.

Stationary in St. Louis

I recently spent several days in a train station in St. Louis avoiding talking to IDS vendors. Well, a retired train station, now a Hyatt Hotel and tourist mall. And the site of yet another security conference, the CSI NetSec. I enjoyed the conference but never entered the exhibit floor.

Sometimes vendors recognize me, and that can spell trouble because some of them think I still write for *UnixWorld*. Most do not recognize me but will happily rope in anyone who comes within range, whether that person needs their product or not. Some of the products are quite good, while others exhibit successful marketing: lots of flash without much substance – but with nontechnical people making the buying decisions, who cares?

And the really funny thing is that I am reminded of something that most of us put in our mouths daily, even though it is poison. Having had a chance to reread the ID workshop summaries and to listen to several presentations on hacking techniques that pass beneath the radar of the current crop of ID products, I find myself comparing ID systems to fluoride toothpaste.

Like toothpaste, ID systems can be very useful. A properly configured system can tell you about what has happened on your network and, in rare circumstances, even do something about some denial-of-service attacks, such as SYN flooding or smurfing. Fluoride toothpaste changes the chemical structure of tooth surfaces, making it more difficult for the acids produced by mouth bacteria to produce caries.

But look closely at the small print the next time you brush your teeth. You will notice the obligatory reference to the Poison Control Center, along with a suggestion that you not swallow the toothpaste, and that more than the amount normally used can be dangerous to children. Fluoride ions, found in most toothpastes, are poisonous, but tolerable in small amounts.

ID systems should also be considered mostly useful, but perhaps dangerous in large amounts. IDSes can tell you a lot about what is happening, or might be happening (the false positives), in your network, the next time you get a chance to look at the IDS. For myself, I do want to log what has been happening, but I am more inclined to follow Peter Neumann's and Ed Amoroso's suggestion: focus first on hardening your systems and infrastructure, and eliminate the targets for attacks.

And don't forget not to swallow.

conference reports

This issue's reports focus on on the First Conference on Network Administration held in Santa Clara, California, April 7-10, 1999, and on the Workshop on Intrusion Detection and Network Monitoring held in Santa Clara, California, April 9-12, 1999.

Our thanks to the summarizers:

David Parter

Ryan Donnelly

David Klotz

Rik Farrow

First Conference on Network Administration

SANTA CLARA, CALIFORNIA

April 7-10, 1999

KEYNOTE ADDRESS

Home, Road, and Work Have Merged Via the Internet

Norm Schryer, AT&T Research

Summary by David Parter

"If the code and the comments disagree, both are probably wrong." – *Norm Schryer*

Norm Schryer has an OC-3 to his house. And a cable modem. Why? He works for AT&T Research, where his group has been investigating broadband network services. In fact, they have spent most of the last five years on the infrastructure required to do research on broadband services.

Schryer discussed many aspects of widespread high-speed multilocation networking and their implications for network managers:

- Services are widely distributed.
- Distributed responsibility among PCs, laptops, palmtops, etc.
- Central fragility: The center of the network is subject to failure.
- Security issues are important and cause problems too.
- "The bigger and better the network, the easier it is to destroy itself."

WAH/WOOH: Work at Home / Work out of Home

Most members of Schryer's group have cable modems at home. All of them work all the time (or close to it) "fixing it, making it, or doing something with it." They are "on net" all the time. This makes for more loyal employees, some of whom move in order to get broadband services. Those who live in areas where cable modems or xDSL are not available are "geographically challenged."

VPNs

For both the home user and the road warrior, Virtual Private Networks (VPNs) are required for security. The VPN is tunnelled over the broadband service (cable modem, xDSL, etc.) using IPSEC. According to Schryer, VPN software clients are very fragile, and VPN hardware is great but expensive. VPN software is like "rolling ball bearings down a razor blade – if any fall off, IPSEC breaks." Several vendors provide VPN software, but it is often usable only on specific versions of the operating systems and, especially with Windows, often breaks when other software or patches are installed.

Their solution is a dedicated computer running Linux, which provides the VPN and IPSEC software. All the other computers at home do not need to be reconfigured in order to use the VPN.

For performance reasons, Schryer strongly recommended against creating a VPN over the commodity Internet. Instead, you should contract with your broadband service provider for a high-speed dedicated link to your corporate network.

At this point, about half of AT&T Research is using its VPN system from home.

Road Warriors

When travelling, most users have a laptop running a Windows operating system and have to use a VPN software client (instead of the "moat," the Linux tunneling system). Their experience is that the Windows IP stack is very fragile, and IPSEC shuts down and locks out the user when it detects what it thinks is an attack. Frequent reinstalls of the laptops result.

Schryer also discussed other challenges for the network administrator when deal-

ing with high-speed connections to the home and road: high-bandwidth applications such as music; universal messaging; PDAs; and wireless links and network telephony, among others.

In summary, he identified the following principles, with the note that if you are horrified at the scope of the problem, that is good.

- Very strong vendor management is needed: insist on standards, interoperability, and reliability; SNMP management of IPSEC is a disaster; "pin your vendor to the wall on reliability"; if they lie, you pay for it.
- 7x24 customer care – with a global network, customers are always on – 7x24 coverage is "dirt cheap" for keeping customers.
- "Do not step in the Microsoft": Microsoft network software is extremely fragile – can break at any installation of software, even nonnetwork software; Windows is a nonmanaged platform; segment what you are doing – keep critical stuff on a separate platform that is managed.

REFEREED PAPERS

Session: Monitoring and Video

Summary by Ryan Donnelly

Driving by the Rear-View Mirror: Managing a Network with Cricket

Jeff Allen, WebTV Networks, Inc.

"Cricket is *not* the same as MRTG" was the theme upon which Jeff Allen began his discussion of the new enterprise network-management tool. Cricket was born out of a need to forecast network growth and plan for expansion – not merely react to significant changes in usage patterns. In addition, Cricket, like its predecessor, provides for instantaneous "is this link up" management features. Also, like MRTG, Cricket is a cgi-based management tool, featuring graphical displays of traffic trends over configurable time periods with comparison to other dates/times. As such, it provides information for long-term analyses of traffic trends on a specific link or group of links.

The system, while appearing MRTG-like, has evolved in numerous ways. An increase in configurability, via a hierarchical configuration tree, allows for data gathering from scripts, SNMP, Perl procedures, etc. The data is obtained by means of a collector, which runs out of cron and stores data in RRD files, later to be interpreted by a graphing program. The RRD/grapher program implements "graphing on demand," which generates results similar to those displayed by MRTG.

In addition to graphing data, Cricket also gathers application and host-based statistics to monitor such events as PVC states, cable modem signal strength, and router CPU load.

Cricket has taken its place as the successor to MRTG. While it can display much of the same information, its hierarchical configuration tree and improved code allow it to perform many more tasks in a more efficient manner. The Cricket home page is <http://www.munitions.com/~jra/cricket>.

Don't Just Talk About the Weather – Manage It! A System for Measuring, Monitoring, and Managing Internet Performance and Connectivity

Cindy Bickerstaff, Ken True, Charles Smothers, Tod Oace, and Jeff Sedayao, Intel Corporation; Clinton Wong, @Home Networks

The Internet Measurement and Control System (IMCS) was developed to provide quantitative measures of Internet performance. IMCS provides such statistics as HTTP `GET` requests and ICMP echo times to flagship Internet sites. It then uses such statistics to delineate process limits for a given data set.

Such limits are obtained by two Perl procedures, `TimeIt` and `Imeter`. `TimeIt` measures the total time to look up a URL, connect, transfer data, and disconnect. Throughout the transfer period it also logs both transfer and error rates. `Imeter` completes the statistics-gathering engine by adding an ICMP echo component. Pings are issued to such strategic locales as root name servers and large Web sites.

The monitoring and alert component of IMCS is Web-driven. As soon as IMCS determines that the process limits have been violated for a specific measure, an "all-in-one" Web page can alert network operations center staff to potential problems on a link and thus allows for a degree of preemptive network management.

To judge by Intel's experience, IMCS can provide a network administrator with rich performance statistics about specific network traffic. But the system faces the problem of process limit setting, by virtue of the lack of a consistent statistical model.

In the future, the authors plan to integrate an expanded set of services to be monitored, such as SMTP and DNS, and also plan to integrate flow data.

Supporting H.323 Video and Voice in an Enterprise Network

Randal Abler and Gail Wells, Georgia Institute of Technology

Randal Abler and Gail Wells have been exploring the possibilities of implementing H.323, a voice- and video-over-IP transport. The H.323 standard allows for the proliferation of two-way videoconferencing over the Internet and other applications such as distance education.

Developed as a LAN protocol, H.323 is UDP-based. As such, extensive H.323 traffic has the capability of overwhelming TCP traffic on a congested link. In trying to check such traffic levels, most H.323 applications contain the ability to limit client bandwidth usage. In addition, a network-based bandwidth-limiting solution, such as MPLS, could be warranted. As evidenced by testing, bandwidth limi-

tation is crucial not only for network-congestion reasons but also for client-performance reasons. Programs such as Microsoft NetMeeting experienced degraded performance when used on a link with a speed inconsistent with the program's bandwidth setting. Testing also showed that conventional modem links do not supply sufficient bandwidth for acceptable realtime video usage. However, with the advent of digital connectivity to the home, the future of H.323 appears bright.

Session: Configuration Management and Security

Summary by David Parter

Network Documentation: A Web-Based Relational Database Approach

Wade Warner and Rajshekhar Sunderraman, Georgia State University

Wade Warner described Georgia State University's use of a relational database to track and document configuration information for its network. An online Web-accessible database was preferable to paper records because the Web interface allows for access from all platforms, and because paper records don't scale and are hard to use.

The implementation that was described is specific to the GSU network, but it is easy to add other types of devices. It makes extensive use of Query by Example (QBE), so that changes in the device types do not require redesign or rewriting of the user interfaces. Currently they are using Oracle, but they have an MSQL implementation freely available.

Items included in the database are installed software, printers, and specific information about the computers (such as amount of memory, serial numbers, and MAC addresses). They track computers by hostname and IP address, which has caused some issues for multi-homed hosts.

Several commercial tools are available for similar tasks, including populating the initial database. All suffer from the same problem: keeping the database current. The GSU group populates the database by extracting information from a variety of existing sources (such as the DNS system), manually entering the data, and using a Web form.

There are three access levels for viewing the data: network admins have all access; the middle layer can query specific network devices and generate preformed reports; and the users can see the availability of certain resources such as printers and software. All access is controlled with a user ID and password.

There were some interesting questions and comments from the audience about using port-scanning and SNMP to gather some of the initial data. (Wade said that they are working on that.) Also, USC had (or has?) a similar system, requiring the MAC address be entered into the database before they will assign an IP address.

Just Type Make! Managing Internet Firewalls Using Make and Other Publicly Available Utilities

Sally Hambridge, Charles Smothers, Tod Oace, and Jeff Sedayao, Intel Corp.

Jeff Sedayao gave a very interesting presentation on the management of Intel's distributed firewall environment, which relies heavily on make and other common publicly available UNIX utilities.

The Intel firewall architecture has the following characteristics:

- a typical screened subnet architecture.
- inner/outer firewall routers with a "DMZ" containing bastion hosts.
- ISP routers on a common network segment outside the "outer" router (not in the DMZ).
- geographically dispersed routers and firewalls, with failover from one firewall complex (DMZ site) to another as needed. (Currently there are eight firewall complexes.)

The bastion hosts need consistent configurations from one firewall complex to another, and this system has to be managed by limited staff.

Their solution is to view all the firewalls as a single distributed system rather than a collection of independent firewalls. A single set of source information is used to construct customized configuration files for each firewall (using make). Each DMZ network segment is described by a series of serializable segments of firewall access lists that can be combined in any order to function correctly. Each set of rules is stored in a separate file. For a given firewall, the constructed ACL is ordered for the best performance (most common rules first) for that particular firewall complex, preserving the required order of individual rules to enforce the desired access policy.

The bastion hosts are handled in a similar manner. Each is an exact copy of its counterparts at other firewall complexes. There is only one configuration for a given function (DNS, Web proxy, etc.). make is used to manage the configurations, and rdist (with ssh) is used to synchronize the bastion hosts with the master.

Their experience with this system has been positive, with the following lessons learned:

- The system is scalable and robust for both support staff and users.
- There is a fundamental need for discipline.
- It is easy to ensure that a correct configuration is installed at all the firewall complexes. (For example, they had the "Melissa virus" check pushed out before their manager asked about it.)
- It is also easy to propagate an error to all sites.

- Admins must be trained to change the master, not the deployed configuration.
- Version control is necessary.
- Using RCS-like tools, one can easily look at changes from the previous configuration to try to identify the cause of a problem.
- rdist can be used to compare the master configuration tree and the installed bastion host.
- The router configurations are reloaded daily to ensure that the most current version is in use.

Future work is planned on automating testing of configurations before pushing to the bastion hosts, and on automating the firewall ACL optimization.

Tricks You Can Do If Your Firewall Is a Bridge

Thomas A. Limoncelli, Lucent Technologies, Bell Labs

Typical firewalls act as routers, even if the routing function is not otherwise necessary. The network on one side of the firewall is one IP network, and the network on the other side is another network. To applications on either end, the firewall looks like a router. Tom Limoncelli described his experiences with a firewall that acts as a bridge (transparent to the applications) instead.

The major advantage of the bridge firewall is that it can be inserted into the network without requiring a configuration change in any other devices. This also reduces the need for extra transit networks between the traditional (nonfirewall) router and the firewall, and allows for quicker installation (without downtime). In addition, Limoncelli argued that the software is simplified by not having to implement routing functions. (Not everyone agreed with this claim.)

According to Limoncelli, his is not a new idea – it was proposed (and perhaps built?) a long time ago, and then the concept died as people focused on router-like firewalls. He did not mention who had done the initial work.

He worked through several scenarios, including replacing a router-like firewall with a bridge-like firewall without downtime:

Step 1: Program the new firewall with the same rules.

Step 2: Insert into the network (which should take a few seconds and one extra network cable) on the local-network side of the existing firewall.

Step 3: Run both firewalls in series. Check the log files on the inner firewall – it should not filter (remove) any traffic at all. (The outer firewall should have eliminated all filtered traffic.)

Step 4: Remove the router-like firewall (removing the transit network). This will require short downtime, as the external router will have to be reconfigured from the transit network to the local network.

Additional rule testing can be added by deploying the bridge firewall first on the transit-network side. The existing firewall should show no packets filtered, as the outer (bridge) firewall should have removed them.

In addition to replacing existing router-like firewalls (or augmenting existing firewalls to provide multiple layers of defense), the bridge-like firewall can be used to add firewalls to existing networks without reconfiguration. For example, it is easy to add a firewall to protect a manager's office systems without disrupting the rest of the network. Unfortunately, it is also easy for someone to add a bridge-like firewall without consulting the network staff, since the firewall is transparent to the IP layer.

Many of the questions focused on details of the Lucent product, although Tom's experience is with the research prototype.

INVITED TALKS

New Challenges and Dangers for the DNS

Jim Reid, Origin TIS-INS

Summary by Ryan Donnelly

According to Jim Reid, the Domain Name System is definitely at an evolutionary crossroads. The introductions of IPv6, dynamic DNS, secure DNS, and Windows 2000 have sent DNS administrators around the globe scrambling to design more effective methods for managing the DNS infrastructure. In doing so they have encountered innumerable issues associated with redesigning DNS.

One of the primary concerns with DNS evolution is its hasty marriage to WINS, the Microsoft DNS-like system for Windows machines. In trying to integrate the two systems, IETF engineers have had to add multiple new records and features to the DNS system in order to make the migration as painless as possible. Two of the primary facilitators of the change are dynamic DNS and secure DNS.

Dynamic DNS is a process by which hosts are authorized to modify a given DNS zone file. This is necessary to continue to support the "plug-and-play" philosophy expected by Windows/WINS users. In implementing this system, engineers have encountered the problems of excessive zone transfers that are due to constant zone updates, and the problem of host authorization. While the excessive zone update problem can be solved by implementing a periodic zone-transfer schedule, the issue of authorization requires a more complicated solution: secure DNS.

Secure DNS is a method by which keys are assigned to hosts that are authorized to modify DNS data. While partially solving the problem of host authentication, the overhead involved in key management and encryption/decryption is substantial. In addition, the storage of such

1,024-bit keys in zone files has the capability of increasing the current zone files by as much as a factor of 10.

Reid concluded his talk by emphasizing the fact that much development work still has to be done, and that nothing is cast in stone. The presentation is available at <http://www.usenix.org/publications/library/proceedings/neta99/invitedtalks.html>.

Problems with World-Wide Networking

Holly Brackett Pease, Digital Isle

Summary by David Parter

Holly Brackett Pease described many of the challenges – technical, political, and logistical – of deploying a worldwide private IP backbone.

Digital Isle provides a worldwide IP backbone for its customers, bypassing the more common but often congested network access points and Internet exchanges. Instead, they contract for peering with the appropriate Internet service providers in a particular country as needed by their customers.

Technical challenges include understanding BGP and the way it is configured by the various ISPs and backbone providers to assure that traffic does in fact flow over the correct carrier and that routes are accepted (as appropriate) by peers and their providers. Another technical challenge is the differing telecommunications standards and terminology: X.21 is not a movie rating (but you still don't want to use it – G.703 is your friend). Actually, X.21 is an International Consultative Committee for Telephone and Telegraph (CCITT) standard defining the interface between packet-type data terminal equipment and a digital-interface unit. G.703 is a physical/electrical interface specification, used for leased lines (E1s, similar to T1s but all digital) in certain European countries and Australia, and supported by router manufacturers such as Cisco.

Logistical challenges include getting the network equipment to the site with the correct interfaces, cables, and everything else that is needed. Once the equipment has been shipped, it is often necessary to pay a hefty value-added tax to get it past customs. Pease pointed out that Fry's does not exist in most parts of the world, and if you can find a supplier for that missing cable, they won't take your credit card – they want a purchase order. (Anixter is a good supplier – they do have worldwide coverage.)

Political issues include local laws covering IP telephony and content. In one case, they had to register their networks as being from that country in order to be allowed to peer with networks in that country – despite being a partner of the largest ISP.

There were several questions:

Q: Are the peering agreements with the major players in each country peering of two equals, or on a payment basis?

A: Pay all partners – this is a technical peering and interchange agreement, not one ISP trading access to another. Digital Isle is providing a premium service for their customers; there is a cost. They don't peer at the local exchanges at all.

Q: Language issues?

A: Most of the ISPs that they have had to deal with have at least one or two staff with excellent English. If not, they use the AT&T international translation service.

Q: Content filtering?

A: Mostly not a problem, since Digital Isle already has content rules, and their customers are Fortune 1000 companies, not ISPs, and have no downstream customers. IP telephony rules vary from country to country, and it is necessary to make sure that the network enforces those rules.

The Little NIC That Could

Christopher J. Wargaski, RMS Business Systems

Summary by Ryan Donnelly

In an era of high-growth enterprise networks, NICs (network information centers) are becoming more and more expensive to run. Chris Wargaski's presentation on a cost-effective NIC encompassed the idea of a "project staffing model." In this model, the NIC is the first point of contact for all IT staff. Owned solely by the networking department, overall responsibility for the NIC is held by a middle-level manager. Below such a manager are the on-call manager, NIC analyst, and on-call analyst.

It is the responsibility of the on-call manager to track problem occurrence and resolution. Additionally, it is the responsibility of the on-call manager to report relevant problem-resolution statistics to upper management.

The NIC analyst is the primary point of contact within the NIC itself. The analyst is primarily responsible for answering phones and issuing trouble tickets. The analyst is also responsible for monitoring the overall health of the network in order to detect network problems preemptively. When a problem is detected, it is the responsibility of the NIC analyst to assign the ticket to the on-call analyst.

The on-call analyst is the technologist responsible for fixing the problem and updating the NIC regarding its status.

In the project staffing model, both the NIC analyst and the on-call analyst are obtained by rotating engineering/technical staff through the NIC, as determined by the NIC manager. The manager must consider extensive input from staff on scheduling conflicts and hold staff meetings on a regular basis to ensure a constant information flow. It should be emphasized to staff that training, while important and highly encouraged, is optional.

The presentation ended with several comments on the topic of NICs as hindrances as opposed to facilitators.

Splitting IP Networks: The 1999 Update

Thomas A. Limoncelli, Lucent Technologies, Bell Labs

Summary by David Parter

After a rap intro, Tom Limoncelli introduced his talk by commenting that when he and his co-authors first wrote the Bell Labs network splitting paper (1997), they were unsure if network splitting was interesting to anyone and if it would ever be done again. He then asked if anyone from Microsoft or Hewlett-Packard was in the audience – since they will (or may be) facing the same issues soon. In fact, the sysadmin group at Bell Labs has since been called upon to renumber more networks, involving more total computers, than in the original work.

The original problem was splitting the Bell Labs research networks (on a computer-by-computer basis) between AT&T and Lucent/Bell Labs, because of the AT&T "tri-vestiture" (splitting into AT&T, Lucent, and NCR). In addition, they took the opportunity to upgrade the networks, reorganize the file servers, and bring the workstations into a consistent configuration.

The project was a major success, mainly as a result of the amount of advance planning and effort spent "getting it right" before widespread implementation.

Prior to actually renumbering the networks, it was necessary to identify which workstations and file servers were destined for which networks. In some cases, this was not known for several months, because of uncertainty about which members of the staff were going to which company. In addition, data stored on the file servers had to be sorted and moved to a file server destined for the correct network. An important tool in this phase was a Web page that allowed users to check on the status of the project and read information about specific resources. Additions and corrections were supplied by the users. The users had less concern about potential disruptions since they were involved in the planning process.

The information about which workstations and file servers were destined for which network was also used to populate new NIS netgroups named for the new networks. The new netgroups were used to control filesystem exports on the newly reconfigured file servers. By generating the netgroups from the master list of which workstations were destined for which network, they eliminated a potential inconsistency.

The key technology used in the actual renumbering project was multi-netting all the Ethernets, so that all the original IP networks and the new networks were on the same physical networks. This allowed them to renumber workstations one at a time without disrupting all other workstations on the network.

Repatching data jacks from their current network hub or switch to a network switch designated for the correct future network was done in parallel with the renumbering. As all the switches and hubs were connected together, this was transparent to the users.

When the workstation and fileserver renumbering was thought to be complete, the connections between the two new (physical) networks was reduced to a single bridge. Traffic was monitored at the bridge in order to detect workstations and file servers that ended up on the wrong physical network. After all cross-traffic (other than broadcast) was eliminated, the bridge was removed, the two networks were independent, and the project was completed.

Network Management on the Cheap

Rob Wargaski, RMS Business Solutions

Summary by Ryan Donnelly

Rob Wargaski's talk on cheap network management included a fairly extensive summary of UNIX network-management power tools and their most effective uses. He began his talk by discussing two different types of network management: tactical and strategic. Tactical network management is essentially day-to-day reactive monitoring and firefighting. Strategic monitoring is more focused on data analysis and trend forecasting.

Some of the tactical tools mentioned included ping, traceroute and arp, used to assess baseline network connectivity. An additional tool is K-Arp-Ski, a comprehensive X-based sniffer and network analyzer. It can monitor an IP/MAC address combination, multiprotocol traffic, and NIC vendor information. He also mentioned several SNMP-based tools, such as SNMPSniff and Sniffit, along with the standard UNIX tcpdump.

Discussing strategic network-management tools, Rob first brought up the topic of enterprise network-management packages such as HP OpenView. He also mentioned some CMU/UCD character-based SNMP tools with API interfaces. These included snmpget, snmpgetnext and snmpwalk. The combination of these three tools allows a network administrator to retrieve either a select amount of data from a network device or the entire tree of device information. Because such data is atomic in nature, strategic management requires postprocessing of the information in order to determine trends and typical network behaviors.

The Evolution of VLAN/ELAN Architecture at Vanderbilt University

John J. Brassil, Vanderbilt University

Summary by David Parter

John Brassil described the Vanderbilt University network and its evolution from a "flat" bridged network to one making use of many virtual LANs (VLANs), mostly grouped by common services or administrative structure. Vanderbilt's network includes almost 10,000 computers and will grow soon with the addition of student dorms.

The talk included a large amount of background information on VLANs, emulated LANs (ELANs), and ATM LAN Emulation (LANE).

The network is continuing to evolve:

- While the IP network is currently not routed, they will switch to a routed IP network soon. This is difficult because in the past they have had a lack of planning in IP address allocation.
- They are encouraging users to use local Internet service providers for off-campus connectivity using cable modems and xDSL, and are phasing out their campuswide modem pool.
- They have recently added a firewall at their Internet connection. It is currently being used only for traffic logging, not as a firewall.

Interoperable Virtual Private Networks (VPNs), Directory Services, and Security

Eric Greenberg, Seine Dynamics

Summary by Ryan Donnelly

Eric Greenberg presented several different uses and implementations of Virtual Private Networks (VPNs). Some potential uses include remote dial-in applications, private networks, and business-to-business secure networks.

There are several methods for implementing such VPNs. The first is the PPTP protocol, which provides for multiprotocol tunneling over PPP. While beneficial for moving such protocols as Appletalk and IPX over an IP link, PPTP does not introduce any new security features. L2TP, however, does.

L2TP was described as the successor to PPTP. It is transport-independent, making it a much more viable option for networks that implement various transports, and does not necessarily require IP. It is also UDP-based and has the added feature of enabling digital certificates.

X.509 certificates are yet another method of implementing VPN security. These allow machines within a network to authenticate one another by means of a unique certificate that is stored on a central directory server. An offshoot of this method is public key cryptography, which can be implemented by sending a copy of a public key with the certificate itself. Such certificates can also be issued by numerous emerging certificate authorities, such as Verisign.

IPSEC is an additional method of designing secure virtual private networks. It provides for IP-only tunnelling and is integrated into IPv6 via the "next-header method."

Greenberg also discussed some possible security-policy and key management issues. He suggested that certificate and key management be performed on a central directory server that would hold the certificates and their associated security policies.

In a VPN, however, security is not the only concern. As traffic levels continue to rise, quality-of-service (QoS) issues also arise. Multiprotocol Label Switching (MPLS) allows VPNs to influence QoS issues by assigning a priority to a certain type of traffic.

Greenberg concluded his talk by analyzing the features of PPTP, L2TP and IPSEC, and the environments in which each is most useful.

Internet Measurements

k. claffy, Cooperative Association for Internet Data Analysis at the San Diego Supercomputer Center, UCSD

Summary by David Parter

k.c. claffy gave an entertaining and blunt talk about Internet measurements, tools, and interesting findings from CAIDA. She also provided a lot of answers to what she called "cocktail party" questions – interesting (but not really useful) facts about the Internet. According to k.c., the current state of measuring the Internet is "abysmal science." Because of the growth of the commercial Internet, researchers can't get access to the network infrastructure and operational data as well as they could when the primary network was NSFnet. They do have access to the vBNS, which helps. (vBNS is very-high-performance Backbone Network Service, a product of a five-year cooperative agreement between MCI and the National Science Foundation. See <www.vbns.net>.)

She pointed out that this doesn't stop researchers from building "junk" and doesn't stop users from doing "random junk." There is a lot of measurement activity now (at least 11 organized groups), but the groups don't talk to one another, and at this time there is no correlation of the data sets (e.g., workload versus traffic).

She identified and described four areas of measurement:

- topology
- workload characterization (passive measurements)
- performance evaluation (active measurements)
- routing (dynamics)

One of the more interesting aspects of her talk was the visualization tools that CAIDA is using to explore the data. Unfortunately, they are hard to describe.

Readers are advised to visit <www.caida.org> and look at the samples. Slides from the talk are online at <http://www.caida.org/Presentations/>.

Some of the "cocktail party" facts:

- Half the packets on the Internet are about 40 bytes long.
- Half the packets on the Internet have a full 1500-byte payload.

Workshop on Intrusion Detection and Network Monitoring

SANTA CLARA, CALIFORNIA

April 9-12, 1999

KEYNOTE ADDRESS

Dr. Peter Neumann, SRI

Summary by Rik Farrow

Dr. Neumann has a long history in both computers and security – his first programming job was in 1953. He was the co-author of the paper that described the file system for MULTICS (an ancestor of UNIX), which included much stronger provisions for security than most operating systems before or after. He worked with Dorothy Denning on IDES, an intrusion-detection project that had its roots in work beginning in 1983. Neumann was also the co-author, with Phil Porras, of the paper that won the workshop's Best Paper award, about EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances).

Neumann spoke from notes instead of giving a prepared speech. His premise is that our problem with security today is a structural one, and we won't have secure networks or computers unless we have secure operating systems, applications, and even programming languages that have support for security. One person, an ex-CERT team member, muttered that this was "the same old stuff." Sure, but until we get this right, we really cannot move on.

"We don't start with secure systems," proclaimed Neumann. Today's intrusion-detection (ID) systems have chosen to attack the easy problem, detecting known patterns of misuse. Today's ID products ignore insider abuse, which – like penetrations – would better be prevented than detected. The need for ID would be greatly reduced if operating systems provided better authentication, stronger cryptographic algorithms and implementations, stronger differential access controls, and, in some cases, multilevel security.

With today's operating systems, there is little hope of maintaining good security. Operating systems are already too unwieldy to administer properly, and some systems can be totally impossible to manage. As an example, he mentioned Microsoft and alluded to the reported 48 million lines of code that will become Windows 2000.

Also, with today's systems it is possible for a single node to contaminate an entire network. As examples, he mentioned the 1980 ARPAnet collapse, the 1990 AT&T long-distance collapse, the AT&T Frame Relay Network outage, and the Galaxy IV satellite outage. (Remember when many pagers suddenly stopped working?)

EMERALD is designed so that it can be integrated with other tools. Correlation of data from many dispersed sensors represents a serious problem. Another paper given in Oakland in May provides more details about how EMERALD copes with this problem and also characterizes the rule-based component. (See <http://www2.csl.sri.com/emerald/pbest-sp99-cr.pdf>.) The use of further reasoning can weed out the false positives found in anomaly-based ID systems.

Neumann concluded by saying that he is trying to convince the US Department of Defense to fund the robustification of an open-source version of UNIX and other good open-source software. "Jails are not the answer to our security problem; robust systems and networks are required as a basis for real security," stated Neumann. So-called software engineers need to be taught principles of software engineering and security. Data abstraction, encapsulation, and language features can help to make it easier to write secure software.

Dr. Neumann left ample time for questions at the end of his remarks. Someone asked if authentication will solve problems, to which he answered, "Cryptographic authentication can help solve the problem." Fixed passwords outlived their usefulness many years ago. The challenge here is to build a much more robust structure coupling authentication with identification. He'd like to see it done sooner rather than later.

The next person asked about problems with differential ACLs (access control lists), and did MULTICS solve the problem? Neumann replied, "No, and the Orange book didn't fix it either." Roles don't do it, and today's OSes have serious vulnerabilities. We should put trustworthiness on trusted servers. We need open source because it can be analyzed.

Dan Geer tied in examples from bacteria defense systems and asked if these are truly relevant to ID. Neumann's answer

was that while the biological model is not really applicable, there are some lessons from biodiversity. Cyberdiversity can help with defense; the Melissa virus shows what can happen in a monoculture environment, which is another violation of the Einsteinian dictum that everything should be as simple as possible but no simpler.

Tom Limoncelli asked about best practice. Neumann answered that you can do simple things, but that alone is not enough. The open-source paradigm has an enormous community working to find problems, but open source does not solve the problem by itself unless the systems are adequately robust. Someone else asked about electronic commerce and whether the rush to insure risk would help. Neumann replied that the risks at the moment are relatively low. He quipped that maybe we need a Chernobyl in the computer world.

You can find out more about Peter Neumann's life, interests, projects, and papers by visiting <http://www.csl.sri.com/neumann/>.

REFEREED PAPERS

Session: Analysis and Large Networks

Summary by David Klotz

Analysis Techniques for Detecting Coordinated Attacks and Probes

John Green and David Marchette, Naval Surface Warfare Center; Stephen Northcutt, Ballistic Missile Defense Organization; Bill Ralph, ATR Corporation

One of the hot topics in intrusion detection is how a coordinated attack can be recognized. David Marchette gave a statistician's viewpoint on this question. By his own admission, the talk gave some clues for how to detect coordinated attacks but offered no magic bullet. He went on to say that as attackers get more sophisticated, we must rely more and more on coincidence to detect coordinated attacks.

A coordinated attack is one attack from two or more computers. It is not a requirement, however, that you can tell for sure that multiple machines are involved. In practice this is often impossible. Essentially, coordination is a symptom or the appearance of a pattern. Marchette gave several examples of coordinated attacks he has seen:

- Coordinated traceroute, which can be used as part of a mechanism for denial of service by taking down a system upstream from yours.
- NetBIOS scans, in which he found that only IP addresses that didn't exist were being probed. One explanation for this was that attackers might have been interested only in new machines, which presumably haven't been secured yet.
- Reset scans, which he was not even sure were attacks. They might have been a naturally occurring event, but they could also be used to do inverse mapping by recording host-unreachable messages generated by the RST packet. TCP hijacking was another possible explanation.

Detection of these types of attacks can sometimes be done by recognizing patterns in audit data, either by looking for anomalies or seeing signatures of known attacks. Unfortunately, very slow attacks that fall outside the window of detection really cannot be recognized.

Someone asked how long a window Marchette uses. His response was that though his team has over half a terabyte of packet header data, it is unlikely they would look at all of it. He estimated his window to be around a couple of days, maybe a little longer.

Someone else mentioned detection of spoofed addressed attacks, saying that if you clustered IP addresses, TTLs, or sequence numbers of scans, then the spoofed ones would stand out. Unfortunately, attack tools that spoof have already become sophisticated enough to escape most detection using this method, and the trend is toward even better tools.

Intrusion Detection and Intrusion Prevention on a Large Network: A Case Study

Tom Dunigan and Greg Hinkel, Oak Ridge National Laboratory

In many situations, intrusion detection can be done in a locked-down system where access is limited and abnormal behavior can be specified rather than inferred. In many scientific centers, however, such restriction is very unpopular and is strongly resisted. Scientific staffs, including offsite collaborators, require easy access to accounts and data, often over unencrypted channels. A further complication is the fact that many users don't see security as an issue until they themselves are victimized. Greg Hinkel presented a paper on detecting intrusions in such an open environment, at Oak Ridge National Lab.

Hinkel described a layered approach taken at ORNL:

1. Firewall: to place some limitations on access.
2. External monitoring: to monitor traffic from outside.
3. Internal monitoring: Honeypots are placed around the network, and some scanning of internal systems is done to look for known vulnerabilities.
4. System administration: System administrators are taught how to set up systems correctly and what things to watch out for.
5. End users: Users are educated about security issues, such as the danger of cleartext passwords and the importance of their accounts.

The hardware configuration involves

extensive monitoring of traffic, both internal and external. For example, all external sessions are keystroke-logged, providing a complete record of what happened. Honeypots are used internally to allow the tracing of intrusions when they do happen. Realtime detection is also complemented by scripts that comb through logs looking for "interesting" things, such as IRC traffic on dedicated scientific computing machines.

Three specific cases from the paper were described. The first involved a Russian attacker who was able to gain root access on one of the SGIs at the lab. They were able to detect the attack by noticing a port scan and then capture the attacker's session with the keystroke logger. A similar attack from Brazil was detected by the same means. A third attack, which involved a cracked password, was detected when the TCP connection logger noticed an unusually high number of connections coming from one machine.

Most of the interest from the audience was in the keystroke logger. One question dealt with how the logs are kept safe from tampering. Hinkel responded that the logs are kept on highly protected machines that can't be reached from the outside and are well hidden. Someone else wondered how ORNL plans to deal with encrypted sessions; this issue hasn't been addressed yet.

An Eye on Network Intruder-Administrator Shootouts

Luc Girardin, UBS, Ubilab

Luc Girardin presented a method to analyze network traffic statistics visually, allowing detection of possible attacks by inspection. Unlike most current IDSs, this system relies on human monitoring and takes advantage of human ability to understand complexity.

Girardin sees current IDSes as systems that rely on implementation of more and more complicated locks to respond to more advanced attacks. In order to try to break out of this loop, a new approach is needed. Rather than combing through network packets looking for anomalies or misuse, traffic data is mapped topographically. In this way similar events are represented as being close together, and dissimilar ones farther apart.

Luc Girardin

By mapping network traffic, this approach takes advantage of the almost universal ability of human beings to comprehend geographic maps. Relationships such as proximity or optimal path apply to network maps in a way similar to how they apply to geographical maps. The actual mapping is done using an unsupervised neural net, which has no a priori knowledge about network traffic patterns.

Session: Software and Processes

Summary by David Parter

On Preventing Intrusions by Process Behavior Monitoring

R. Sekar, Iowa State University; T. Bowen and M. Segal, Bellcore

This paper presented an active approach to preventing intrusions. According to the authors, damage can happen only as a result of system calls by modified programs and network packets delivered to the target host. Damage occurs when programs deviate from the intended behavior. The authors' damage-prevention technique is to model the correct behavior of a program in terms of system calls and detect any deviation from the model. They did a paper-and-pencil analysis of 96 CERT advisories and determined that most of the program deviations were detectable with their method.

There are two parts to their system: an offline preparation stage and a runtime monitor. The offline stage generates the detection engine on the basis of the behavioral specification of a program. The runtime system uses a kernel-level system call interpreter for Linux, which intercepts system calls just before and immediately after the kernel implementation of the system call. The detection engine compares the system-call sequence with the expected sequence. Any deviations are detected, and execution is prevented.

In response to a question about code availability, it was stated that their work is currently a prototype of the concept.

Intrusion Detection Through Dynamic Software Measurement

Sebastian Elbaum and John C. Munson, University of Idaho

Sebastian Elbaum, a student at the University of Idaho, presented his work on intrusion detection "from the inside out" of a program. (He received the Best Student Paper Award).

Most tools use standard audit trails, which include information only about points for which audit records are generated. By looking at the software "from the inside out," the authors expect more low-level detailed information, which will make it easier to detect abnormalities.

Their method is to model the expected behavior of the program as a series of function calls. In practice this is an extended call graph and execution profile of the program, and the probability distributions for sequences of operations.

The implementation is to add software instrumentation to the program source

code; in this case, the Linux kernel. At this point, they have successfully instrumented the kernel and produced a nominal profile. Changes in user behavior (for example, students on spring break instead of in class) have led to false positives, and more work is needed on developing the nominal profile.

Learning Program Behavior Profiles for Intrusion Detection

Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz, Reliable Software Technologies Corp.

Anup Ghosh observed that looking at process behavior, instead of user behavior, is a shift in intrusion-detection methods, and that it was nice to see several papers on this at the workshop.

The advantage of anomaly detection, as opposed to attack-signature detection, is the possibility of detecting new (as yet unknown) attacks. This is an important goal of the authors' work. Their premise is that abnormal program behavior is a primary indication of program or system misuse.

Like the previous presenters in this session, they construct profiles of expected program behavior. Their work builds on previous "computational immunology" work at UNM, which looked at system-call sequences, and at Columbia, which used data-mining techniques on the UNM data.

Their implementation uses available technology such as the Sun Basic Security Module (BSM) auditing facility and the Linux strace program. With Sun's BSM, programs typically create 10-20 different BSM events (out of about 200 possible events).

He discussed three algorithms for using the profile data: equality matching, like that done by UNM; back-propagation neural networks; and recurrent networks. Each algorithm was tested using known attack traces. A "Receiver Operating Characteristic" (ROC) curve is used to plot the effectiveness of the method with different settings. The Y-axis represents the probability of detecting an attack, the X-axis the probability of reporting a false positive. At (0,0) nothing is reported. At (1,1) all attacks are reported; however, 100 percent false positives are also reported. (0,1) is considered an oracle: 100 percent correct positives reported, with no false positives.

Table Lookup

Table lookup (equality matching) is the baseline for comparing the methods, since it is the easiest to implement and is known to be fairly good at anomaly detection. The profile consists of fixed-sized "windows" of events and their frequencies. It is then analyzed at various granularities (individual windows, clusters of windows of various sizes, and over the entire session), counting the number of anomalies found. For the detection phase, tunable parameters include how many anomalies at each level will trigger a report, size of the windows, and amount of training the system undergoes.

The table-lookup method is simple to implement and good at detecting new attacks, but the false-positive rate proved to be too high. And this method does not generalize – it is based entirely on memorizing expected patterns.

Neural Networks for Intrusion Detection

Neural networks learn the normal behavior by observation, and they provide the ability to generalize from past behavior. They can detect deviations from normal behavior. The authors implemented a "back-propagation" neural network with supervision. It proved to be suitable, but it did not do as well as equality. They concluded that this is because both training and tuning the network are difficult. They also noted that overtraining the network leads to a pure memorization approach.

Ellman Networks

Their third approach used an Ellman network, which is similar to the network used in the previous approach but includes state information. The performance was remarkably better: 0 percent false positives up to nearly a level of 80 percent correct positives — but so far, only on the one set of data that they have tested. They are continuing to investigate this approach to real time intrusion-detection systems.

Session: IDS Systems

Summary by David Klotz

Automated Intrusion Detection Methods Using NFR: Methods and Experiences

Wenke Lee, Christopher T. Park, and Salvatore J. Stolfo, Columbia University

As attackers and attacks have grown more sophisticated, intrusion-detection systems have had to be brought up to speed. Often the rules used to detect attacks are hand-encoded, and it can be a laborious process to keep these systems current. Christopher Park presented work that

Christopher Park & Anup Ghosh

represents a new approach to discovering and encoding rules into an IDS.

After giving the obligatory description of anomaly and misuse detection, Park went on to describe his group's system, which uses data mining to discover frequent patterns in connection data. These patterns are used to come up with machine-learned guidelines, using RIPPER (a rule-

learning system), and then encoded into Network Flight Recorder (NFR). As with most other machine-learning systems, a training period is required.

Several questions followed the talk. One attendee wanted to know if they had looked at other sniffer or detection systems besides NFR. Park, who earlier in the talk had indicated that NFR had been chosen because of its extensibility, real-time alert capability, and noninterference with network traffic, replied that they had looked at several software packages and had all agreed that NFR best met their needs. He reiterated that its extensibility was what made it the most useful for their research.

Someone else asked what features are looked at during rule creation. Currently only features that can be gleaned from packet headers are used. Do they plan to look at data rather than just header information? Park responded no, they only look at headers. A response to another question indicated that as bandwidth increased, the rules discovered also changed, specifically those that deal with fragmented packets; however, no data had been taken that would show whether or not accuracy went up or down with increased traffic.

Experience with EMERALD to Date

Phillip A. Porras and Peter G. Neumann, SRI International

This paper received the Best Paper Award. Phil Porras presented an overview of the EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) intrusion-detection system. The EMERALD approach has been to shift analysis from centralized, monolithic intrusion detection to distributed, lightweight sensors that can be deployed strategically. Though each module would gather data locally, they would feed information to one another in order to get a global view. There is also a standard API that allows integration of third-party components into the EMERALD system.

EMERALD uses both anomaly detection and an expert system to recognize known attacks. The anomaly detection builds a profile of normal activity and compares both short- and long-term patterns. It monitors current activity and sets off alarms when it departs from the profile of what is normal. The expert system fully integrates a P-BEST shell. Rules are translated and compiled directly into code modules. Hence, no interpretation is done, so the rules can be run very quickly. Porras emphasized that creating EMERALD modules was not difficult and mentioned a graduate-school class where the students built their own modules, using P-BEST, in about two weeks. He also mentioned that the EMERALD team is currently working on a Web-based development environment.

EMERALD also attempts to correlate activity across modules. Equivalence recognition is done to determine when two reports refer to the same attack. Commonalities between reports are also looked for, as well as vulnerability interdependencies across physically distributed components. Commonly seen sequences across domains are also noted. One attendee wanted to know how EMERALD handles system calls. Porras responded that it traps Solaris system calls, and that they are working on something similar for FreeBSD, but that they had nothing for Linux. In response to another question, Porras mentioned that he does not believe there is any IDS on the market that can be compared to EMERALD.

Defending Against the Wily Surfer – Web-based Attacks and Defenses

Daniel V. Klein, Cybertainment, Inc.

Dan Klein started his talk by answering the question of the hour: yes, his mom – and in fact most of his relatives – knows what he does for a living. For those neither related to him nor not present at his talk, Klein is the technical person at Cybertainment, Inc., one of the larger Internet adult-entertainment providers. From this vantage point, he has a good view of a large variety of Web-focused network attacks.

Klein started his talk with a prediction that the nature of the Web will change tremendously in the near future. Now almost everything is free, except for porn. He believes this will change when people realize that the costs associated with going on the Web are not covered by ad revenues. He also pointed out that the adult-entertainment industry is what drives the Web, since it is that rare commodity that people are willing to pay for. People are also willing to steal it.

Klein broke Web-based attacks down into three categories: simple theft, breaking and entering, and felonious assault. Simple theft refers to actions like registering common misspellings of a popular domain name (like netscpae.com), or registering variations on top-level domains (such as whitehouse.com). While these actions aren't in themselves illegal, these alter-sites could easily be used to fool users into divulging credit-card or personal information.

Breaking and entering refers to actions such as domain-name stealing or password cracking. Several domain-name controversies have been mentioned in the media recently, which only highlights how easy that particular attack is. Felonious assault covers such things as DNS-cache poisoning or JavaScript frame spoofing.

One of the most common problems that adult sites face is password sharing. One person will get a hold of a password, by legitimate means or otherwise, and then post it to a public place. There are in fact whole Web sites devoted to this. Of course this makes password sharing fairly easy to detect, since huge traffic spikes occur shortly after one has been posted. Klein uses software to check for these spikes twice a day and disables the account when it's found.

One novel approach that some sites use to turn this attack to their advantage is to post fake usernames and passwords, which redirect to a click-through ad that generates revenue. Someone raised the question of whether or not this kind of activity is itself illegal. Klein argued that it is not, but simply a legal means of taking advantage of illegal behavior.

Another tricky legal area Klein covered was bandwidth theft. If a site downloads images from another site and posts them, this is clearly prosecutable theft, but if a site simply puts links to images owned by others, surrounded by their own advertising, then they have done nothing illegal. Meanwhile the other site is the one paying the bandwidth hit for serving the images, while the one using the links is able to generate the revenue.

Clickbots represent another form of illegal activity. They are designed to generate hits on click-through ads that produce revenue on a per-click basis. These can be very hard to detect if done well. One person wanted to know when these were considered illegal rather than just immoral. Klein responded that in all cases he considered them immoral, but clickbots that generate fake hits that actually create real revenue could be legally classified as fraudulent. A trickier situation is one where the clickbot is generating hits that move the site up on a top-ten list. Here there is no direct revenue generated, and in fact it can be argued that the top-ten list benefits too.

The talk ended with Klein giving some advice on how to go about protecting yourself: try to think like a bad guy, and most of all have fun; remember that we too can be devious.

Session: Network Data Processing and Storage

Summary by David Parter

Preprocessor Algorithm for Network Management Codebook

Minaxi Gupta and Mani Subramanian, Georgia Institute of Technology

Minaxi Gupta presented her work on preprocessing a "codebook" for network management. The codebook approach is to try to identify all possible causes for every possible failure in the network. The symptoms and problems are put into a matrix, whereby one can easily determine the problem for a given symptom. A well-formed matrix eliminates redundant entries so that each symptom leads to a specific root cause of the problem.

Once a codebook has been created, an automated system can use the codebook to identify problems based on the symptoms (observed failures). Gupta's work is a technique to ensure a minimal codebook, which is crucial to an efficient runtime system. In her presentation, she detailed the mathematical aspects of the preprocessor technique.

A question from the audience challenged her assumption that an optimal codebook could be produced and was in fact useful, since most systems have multiple errors and it is impossible to build a perfect system where everything functions exactly according to plan. It was strongly recommended that she consider using redundancy in the codebook. Gupta answered that this was a good observation, and that they would be adding known probabilist symptoms, which should address this concern.

The Packet Vault: Secure Storage of Network Data

C. J. Antonelli, M. Undy, and P. Honeyman, University of Michigan

C. J. Antonelli described work on secure storage of captured network packets. The premise is that recording only the headers or some other subset of the total data will cause some information to be lost that could later prove valuable in analysis, in intrusion detection, or as evidence. If this data is to be kept, it must be secured in a manner that allows release of selected traffic while continuing to protect the other traffic.

The architectural goals of the project include: use of commodity hardware and software, completeness of the packet capture, permanency of the record, and security of the record.

The packet vault utilizes two (commodity) Pentium workstations, one running OpenBSD, the other running Linux. The collector workstation runs OpenBSD and accumulates packets from the network, using a modified Berkeley Packet Filter (BPF) to write directly to a memory-based file system. Each packet is encrypted and then transferred via RPC to the archive station (running Linux). The archive station creates a filesystem image of the encrypted packets and metadata, then records a CD of the image.

The archived data is organized as follows: The source and destination of each packet is obscured by means of a translation table, which is encrypted using a "translation table key" that is changed for each CD. The payload of each packet is encrypted using a "conversation key," which is unique for each (source, destination) pair on the CD. The "conversation key" is derived from the volume (CD) master key and the packet headers. All the keys for a given volume are stored on the CD, using PGP encryption, where the private master key is held in escrow (and is not on the archiver system). In this case, the master key would be held by the University of Michigan Regents.

The encryptions are done using DESX, a modified DES cipher that is believed to be equivalent to 95-bit keys for this application.

While the system worked as a prototype, they have only limited experience with it. They were able to use it for two weeks on a 10Mbit research Ethernet, storing 8GB on 15 CDs. One bottleneck was the CD-ROM recorder, which has to be run at no more than 2X recording speed. The biggest challenges, however, are administrative.

The talk identified the following outstanding issues:

- Limits of DES: With newer CPUs, 3DES should be feasible.
- Limits of passive analysis: There are several known limits to passive analysis for intrusion detection. The packet vault is immune to time-to-live (TTL) tricks, since it does not do packet reassembly, but passes on all packets to the destination undisturbed. An intrusion-detection system using the data from the packet vault would have to address this issue.
- Evidence handling: The goal of the packet vault is to "freeze" the scene of the crime, but steps must be taken to assure continuity of the evidence. Partial solutions include adding digital signatures to the CDs (not using the escrowed private key) and assuring that the procedures are auditable. However, this just pushes the problem back a level to trusting the initial packet-collection system. In court cases, it is often a "competition of gray areas" between the prosecution and the defense; rarely is the evidence black and white. It is believed that adding digital signatures and audits will improve the chances of a jury accepting the evidence.
- Legal issues: There are a lot of potential legal issues with the packet vault, especially for a university. Among them are: carrier-transport / ECPA (Electronic Communications Privacy Act); student information / FERPA (Family Educational Rights and Privacy Act); privacy and First Amendment concerns; human subject guidelines; ownership and copyright issues; right to know / Freedom of Information Act; discovery and evidence rules; search-and-seizure rules; civil liability.

In order to better understand the legal issues, the authors commissioned a six-month study by a law student. He concluded that there is little case law to provide guidance on these issues, and that "fishing trips" for data are possible under the Freedom of Information Act.

For these reasons, the University of Michigan will not be using the packet vault at any time soon. The authors believe that there are fewer such issues in corporate private network environments.

Future work is needed on the intrusion-detection system using the captured packets, better cryptography, digital signatures, the administrative interfaces, and keeping up with faster networks.

Real-time Intrusion Detection and Suppression in ATM Networks

R. Bettati, W. Zhao, and D. Teodor, Texas A&M University

R. Bettati presented work on a real-time intrusion-detection and suppression system. It is targeted to a distributed mission-critical system using a high-speed switched LAN in a closed environment with high operational knowledge. (An example of such a system is a naval battleship-control system.)

These systems have stringent real-time requirements, such as timing and guaranteed bandwidth and delay. They are vulnerable to denial-of-service attacks such as network topology changes, rogue applications, and "high-level jamming" by unexpected traffic. It is important in these situations to detect an intrusion before a denial-of-service attack happens.

In this closed environment, a naive intrusion is easy to detect. The authors propose building low-level ATM security devices that suppress unknown connections (blocking creation of new ATM VPIs and VCIs) to handle naive intrusions. They can also use these low-level devices to detect violations in a given application's traffic signature. The real-time nature of the applications helps, since the traffic characteristics are well specified and well known.

Session: Statistics and Anomalies

Summary by David Klotz

A Statistical Method for Profiling Network Traffic

David Marchette, Naval Surface Warfare Center

The task of monitoring a large network is complicated by the sheer volume of packets that travel across it. Most of this traffic is uninteresting, but in order to find the packets that are unusual and that may be part of an attack, you have to wade through vast quantities of normal packets. David Marchette's second talk of the workshop presented a statistical method for filtering out normal traffic so that abnormal traffic can be focused on.

The two questions that were addressed were:

- Can normal packets be filtered out while still retaining the attack?
- Do machines cluster according to activity, and can that then be called normal?

Though other packet fields could be analyzed, this work looked only at destination ports. Counts of port accesses were used to look for unusual behavior. Obviously, counts here have meaning only in relation to historical data, which is assumed to be normal. The counts are then looked at on a per-time-period basis and are used to create probability vectors for each port. From this, improbable traffic can be discovered and further analyzed. At this point someone in the audience asked whether the system could detect unusually low amounts of traffic, and whether they would consider this to

be suspicious. Marchette responded that he didn't feel this could be an attack in and of itself, but that it might be an indication. Either way, their system didn't deal with unusually low amounts of traffic.

Unfortunately, this method does not scale well by itself, so in order to deal with large numbers of ports, a simplification is used. Clustering is employed to take care of networks with large numbers of machines. In most servers today, ports range into the tens of thousands. The simplification was to count ports 1 to 1024 individually, but to lump higher-numbered ports together into one "big port" category. An alternative would allow groups within the big-port range to be counted individually, and lump the rest together.

In order to cluster machines, each individual has a port-access probability vector created for it. Euclidean distance is then computed for all vectors and either the k-means clustering or the ADC clustering algorithm is applied. The resultant vector of the cluster is then used for filtering. After running experiments which involved 1.7 million packets and 27 identified attacks, ADC clustering was able to identify all 27 while filtering out 91 percent of all packets. K-means performed slightly worse, filtering out about 76 percent before recognizing all 27 attacks.

Transaction-based Anomaly Detection

Roland Büschkes and Mark Borning, Aachen University of Technology; Dogan Kesdogan, o.tel.o communications GmbH & Co.

In one of the more novel talks of the workshop, Roland Büschkes presented work that applied concepts from database theory to anomaly detection. Like specification-based anomaly detection, transaction-based anomaly detection formally specifies user behavior and then represents protocols as finite-state machine transitions. The protocols are used to define the valid transitions. Each transition is then checked against four "ACID" principles:

- Atomicity: All operations of a transaction must be completed.
- Consistency: A transaction takes the system from one consistent state to another.
- Isolation: Each transaction performs without interference from other transactions.
- Durability: After a transaction finishes, a permanent record is stored.

During implementation, an audit stream of TCP packets is used as input. The stream is sent to a splitter that distributes the packets to the appropriate deterministic finite-state machine. These DFSMs test the atomicity of the transaction. From here the stream is sent to a consistency checker to make sure the transaction leaves the system in a consistent state. Finally, all transactions are sent to an isolation checker which ensures that no transaction interferes with another. The issue of durability, whose meaning is not immediately clear in the context of intrusion detection, was left mostly undiscussed.

A few examples were given to show that network attacks do in fact map to database transactions. SYN flood was shown to violate atomicity and the ping-of-death attack to violate consistency. The talk finished with the promise of an implementation of the system in late June and further investigation of analogies that can be made between database theory and intrusion detection.

INVITED TALKS

Why Monitoring Mobile Code Is Harder Than It Sounds

Gary McGraw, Reliable Software Technologies

Summary by David Klotz


Gary McGraw

Mobile code has been in the news quite a bit over the past few years. First portrayed as "the next big thing" in the form of Java and, later, JavaScript, it soon became clear, as malicious applets and macro viruses began to make the rounds, that it has a downside too. Gary McGraw addressed some of these issues and what is being done to combat them.

Mobile code has grown in importance with the explosion of the Internet. In trying to build a world where light switches and shoes all have IP addresses, mobile code becomes desirable, since you don't want to have to code each device individually. However, McGraw pointed out, the problem of malicious mobile code is not new. In the 1980s, downloading executables was declared "a bad idea." Given that mobile code is both useful and dangerous, how can we use mobile code without selling the farm?

To answer the question, McGraw began by going through examples of malicious activity that can be carried out by several popular forms of mobile code. He looked at JavaScript first.

JavaScript, which has nothing to do with Java and was originally called LiveScript, allows code to be placed directly inside the browser. JavaScript can be used to track Web locations a user visits, steal files from the user's machine, or create a denial of service by redirecting the browser. It can also construct Java applet tags on the fly. This creates a way to

sneak applets past a firewall, since most firewalls deny Java applets by filtering on the <applet> tag before it actually reaches the browser.

A more sinister attack is Web spoofing. This classic man-in-the-middle attack can be used to steal control of a user's view of the Web. As links sent from the intended page are sent back to the user, they can be changed to point back to another site. In this way a user's credit card or personal data might end up going somewhere very different from where they intended it to go. Even secure sockets don't provide a guarantee of security.

As McGraw pointed out, just because the little lock on the browser lights up doesn't mean that the secure connection is going where you expect it to. He urged people to actually check out the certificates on secure sites, saying they might be very surprised sometimes. Attackers can get certificates too.

Another class of malicious mobile code is the macro virus. The Word macro virus was the first of the well known, and well distributed, of these. It was so well distributed, in fact, that when McGraw's first book on Java security came back from the publisher, it came as an infected Word template. While some macro viruses can be thwarted by simply turning off macro execution, at least one – the Russian New Year virus – used an autoexecute feature that could not be disabled. To make matters worse, we have now reached a point where programming skill isn't even necessary to create malicious mobile code. This point was brought home when, not two weeks before the workshop, Melissa – a macro created mostly by cut and paste – became the fastest-spreading virus ever.

McGraw discussed ActiveX next. ActiveX uses Microsoft's Authenticode method for guaranteeing safety. This strategy is faulty on at least two fronts. First, when an ActiveX module is downloaded, a signer is presented to the user and given the opportunity to deny to the module the ability to run. Once the OK is given, though, the module is free to do whatever it wants. McGraw used the analogy of allowing somebody into your office under the pretense of carrying out some task, then giving them free and easy access to all of your records regardless of whether or not they are relevant to the task at hand. Since no sandboxing is done in ActiveX, modules have access to the entire machine. Second, the very foundation of the "signed is OK" philosophy is faulty. The Exploder Control, which was a signed and verified ActiveX module, performed a clean shutdown of Windows 95 when it was downloaded.

Finally, McGraw looked at malicious Java. Though Java has been built with security in mind, attack applets can be built. Responding to the point that some malicious applets have been found in the wild, but no attack applets have, McGraw chalked this up to pure luck and presented a long list of attack applets that have been created in labs. Further, while the latest version of Java has several new security features, problems always crop up with new code.

The philosophy behind the Java architecture has been "add as much as you can while managing the risks." The Java Virtual Machine is used as a guardian, protecting operating-system calls by controlling the entry points. Programs that are trusted are given access to these calls, while untrusted code is denied. In JDK 1.0.2 a black-and-white security model was used, where code was either trusted or untrusted. Code verification was done using type checking, but since this couldn't be done statically, it led to vulnerabilities. The most prevalent attack in 1.0.2 involved throwing an exception in a certain state that caused the virtual machine to confuse types. JDK 1.1 added a sandbox and a signing mechanism that would allow code out of the sandbox. The crypto API was also introduced. 1.1, however, still used the black-and-white security model. With Java 2, a new "shades-of-gray" security model has been introduced whereby individual classes can be assigned different levels of trust.

The talk finished with a look at how to protect against malicious code. Three places for stopping mobile code, in order from worst to best, are the firewall, the client, and the browser. Stopping mobile code at the firewall allows for centralized control but is useless when cryptography is employed. Using the client gives you more control over the environment but is difficult to manage, since often there are a huge number of client machines. The browser seems to be the most logical place, since that is where the running and sandboxing actually take place, but is also tough to manage for the same reasons clients are. He also pointed out that finding mobile code is a nontrivial problem. With cryptography and multiple points of entry for mobile code, it can be nearly impossible to track it down until it is too late.

One method of stopping mobile code before it arrives is blacklisting. Though cheap to implement, it is fairly easy to thwart. Further, if it is done dynamically, it can easily lead to a denial-of-service attack as the bad code database is filled up. Another method, stopping errant applets once they start, is much harder in practice than it sounds. In fact, it can be impossible to stop a malicious thread. A thread is stopped by throwing a `ThreadDeath` exception. If the applet catches and handles this, there is no way to kill the thread. A more devious way to keep the malicious applet running is to put the rerun instruction in the finalizer, which means it will be restarted as a garbage-collector thread. The best approach to stopping a hostile applet, then, is to stop the virtual machine. Finally, policy management can be used, but creating and enforcing a fine-grained policy is very difficult.

In summary, McGraw stated some lessons he's learned from dealing with malicious code in the "trenches":

- Type safety is not enough.
- Real security is more difficult than it sounds.
- It is impossible to separate implementation errors from design problems.
- New features add new holes.
- Humans are an essential element to consider.

During the question and answer period, the topic of remote method invocation came up. How does access checking and transitive trust fit in with RMI? McGraw feels that RMI is problematic, since it's not clear whose access policy should apply. Java servlets also pose a problem, because you can declare yourself a certificate authority. When asked his opinion on the Pentium III processor ID issue, he said that not only was it a bad idea from a privacy standpoint, but because it didn't use crypto it was essentially useless. Another question was whether net security was being reinvented with the virtual machine, and whether we might not see intrusion-detection systems for VMs. McGraw feels that this might be the case and said that stack inspection, which is used in code verification, is similar to an IDS. He also criticized the "penetrate-and-patch" mentality of software vendors that is so prevalent today, suggesting that they should do things right the first time.

Design and Integration Principles for Large-Scale Infrastructure Protection

Edward Amoroso, AT&T Labs

Summary by Rik Farrow

Edward Amoroso gave a cynical, interesting, yet rambling talk about the futility of protecting infrastructure. This is not so surprising from a man who worked on the Strategic Defense Initiative ("Star Wars"), which he called a "silly idea, to think you can shoot down ballistic missiles." SDI wasn't possible in the 1980s, and attempts to knock down even the occasional missile have mostly failed in the 1990s as well. In fact, AT&T does better than that.

For example, AT&T profiles the 230 million calls its communications network sees each day. 30,000 people make their first international call each day as well, which creates a "dot" on your account. If there are a number of these calls, you will get a call asking you if you are making these calls, in case someone has misappropriated your phone or calling card. Amoroso noted that the more money you pay, the sooner you get the call.

ID does provides more information to the network manager. But does this really aid in making a network more secure? To start with, you must be able to protect one little node first. If you can protect a single node, does protecting many nodes provide additive protection? Perhaps, said Amoroso. Maybe one police officer on the corner doesn't deter crime, but a whole bunch standing there might.

Then there is the question of whether to use passive network monitoring, as seen in many popular ID products, or active monitoring, where you touch the operating system by installing OS modifications to monitor for intrusions. Amoroso suggested that you put your ID as close to your most important servers/services as possible. He called this the "Corleone Principle," from *The Godfather.* (You watch those nearest to you very closely indeed.)

A malicious outsider could use any protocol, primarily IP, but also SS7 and C7 (AT&T protocols). Gateways (firewalls) include weaknesses in both design and configuration. ID at least gives you a chance to see what is happening in your network. But the firewall is often the least of your problems.

Always keep your war dialer in your back pocket if you do penetration testing, said Amoroso. If you can't find any other way into the target network, modems will always get you in. On the other hand, put a modem on your honeypot so you can detect war dialing. Create some interesting environment for the call, such as the `login:` prompt.

You can also analyze the output of call-detail logs if you have almost any PBX system and look for war-dialing patterns, such as sequential, short calls from the same originating number. And you probably will not find any commercial product that will do this for you.

NetRanger, now Cisco-owned, but once the Wheelgroup, was wonderful at one point. Now most of the people there have quit. Someone within AT&T bought a copy of NetRanger but did not install it for several months. When they got around to it, they discovered modules missing from the CD that prevented it from working. When they contacted Cisco, they did not know about it either – which implies that a lot of purchased copies of it must not be installed.

Infrastructure may be defined as a lot of stuff that you don't control. So when you are deploying for infrastructure, you are working with stuff you do not control.

Hierarchical handling of events/alerts makes much more sense than isolated, peer processing. And the only tool Amoroso has ever seen that addresses hierarchical processing is EMERALD. EMERALD handles a lot of sensors of different types. NFR does provide you with a toolkit, so you could potentially do this with it as well.

What about response to an ID event? Suppose you have 80,000 operators that can respond to trouble tickets. Can these operators do stuff related to ID on the basis of the alarms generated? Amoroso's answer is no, unless it is to dial someone else's pager.

Remember that IDSes will not address the one thing that most hackers will do to get into your network – use a modem.

Most IDSes address only IP network attacks. So, instead of having one IDS, put them everyplace. If an IDS has fundamental limitations in one place, does putting them everyplace really makes things work better? Perhaps the way to find out is to deploy them and see.

Then you wind up with another problem: disparate information, large volumes of information, difficulty in correlation, responses that may be confused, and (currently) no standards for logging. One solution is to centralize all logs in one place. (The Air Force Information Warfare Center has crews looking at logs.)

Good IDSes can correlate events, and you need a cache to do this. This requires a knowledge base (which most IDSes have today). GUIs hide a basic truth – doing ID or network management is hard, and we do not need someone telling us what we need to see. Not that it is not useful to visualize data. For example, use scatter plots to pick out event patterns. GUIs are good for continuity of funding. NFR is the only one with flexibility; NetRanger only provides an ability to search for string patterns.

Presidential Directive PDD-63, "Protecting America's Critical Infrastructures," calls for important assets to be protected at a macro level. But within the infrastructure, what do you protect? Network control points are essential points in telecom (and thus are natural points to defend). A side note: AT&T handles 60 percent of the telecommunications traffic in the US. To decide what you need to protect, you must identify assets and come up with a coherent architecture.

For ID systems to succeed, they must address some major unsolved problems:

- connecting alarms to realistic response systems
- profiling networks and systems reasonably
- network managers do more profiling than ID researchers do
- correlating information (in-band and out-of-band)
- integrating incompatible audit logs
- filtering massive false-alarm streams
- visualizations for demonstration and analysis

After the talk, someone asked how people get in with modems. Amoroso said that PC Anywhere without passwords makes this simple: Peter Shipley dialed every number in the Bay Area. Emmanuel Goldstein and Phiber Optik have radio shows in NYC that have RealAudio logs – listen to what they have to say about modems.

Jim Duncan suggested looking for slopes in scatter plots, or using Cheswick's algorithm to do your scatter plot along with color to help with the discrimination. Amoroso replied that you can try any combination of IP packet values and look for patterns in the scatter plot – if you can build a scatter plot.

Amoroso ended his talk with yet another story. The cable company televising a heavyweight title fight put up a scrambled banner that only people pirating the channel could see, saying to call an 800 number to get a free T-shirt. They got hundreds of calls. An interesting idea for a honeypot, although somewhat useless for most of our networks.

Experiences Learned from Bro

Vern Paxson, Lawrence Berkeley National Labs

Summary by David Klotz

Vern Paxson's talk was essentially a presentation of his paper on Bro, the intrusion-detection system. He first stated the design goals: high-speed and large-volume monitoring capability, no packets dropped during filtering, real-time notification, separation of system design and monitoring policy, extensibility of the system, and ability to handle attacks on the IDS itself. The eventual design was one of a layered system, with lower layers handling the greatest amount of data and the higher ones doing the most processing.

The event engine is the heart of Bro, which does generic (nonpolicy) analysis. Events are generated by traffic the event engine sees and are queued to be handled by event handlers. Extending the engine is relatively simple, since it has been implemented as a C++ class hierarchy. New classes can be written for new types of events. Event handlers, written in the Bro language, are used to implement a security policy. If an event handler isn't created for a specific type of event, that event is ignored.

The Bro language was designed with the goal in mind of "avoiding simple mistakes." The language is strongly typed, allowing type inconsistencies to be discovered at compile time. One interesting characteristic of the language is the absence of loops. This was done because of the need for speed in processing and the desire to avoid possibly unending procedures. Recursion, however, is allowed.

Two responses were added to Bro recently. One is a reset tool that terminates the local end of a TCP connection. This can be somewhat tricky on TCP-stack implementations that insist on exact sequence numbers; it is done by alternating data and RST packets using a brute-force approach. The second is a drop-connectivity script that talks to a border router and tells it to drop remote traffic from a given site. This script has minimized scans considerably without causing a denial of service up to this point.

After going over Bro, Paxson spent some time explaining why he feels the whole concept of intrusion detection is doomed to failure. Attacks on the monitor, such as overloading it with too much traffic or

using software faults to bring the monitor down, can be defended against, but that still leaves the problem of "crud" that looks like an attack but isn't one. Some odd-looking but legitimate traffic includes:

- storms of FIN or RST packets
- fragmented packets with the don't-fragment flag set
- legitimate tiny fragments
- data that is different when retransmitted

As attack tools get more sophisticated, they will begin to take advantage of the fact that it is often impossible to tell attacks from "crud." Paxson ended his talk by saying that he feels network intrusion detection is a dinosaur, and that host-based intrusion detection is where it's at.

A wine track?

Now, how does that go again? "There was a man from . . ."

Marcus Ranum modeling his new shirt

cisco flow logs and intrusion detection at the ohio state university

by Steve Romig

Steve Romig leads the Ohio State University Incident Response Team, which provides incident response assistance, training, consulting, and security-auditing services. He also works with Central Ohio businesses to improve Internet-security response and practices.

<romig@net.ohio-state.edu>

Mark Fullmer,

and Suresh Ramachandran

Suresh Ramachandran is in the master's program in computer and information science at Ohio State University. His computing interests include networks and Web-based information systems.

<ramachandran.10@osu.edu>

Many Cisco routers support NetFlow accounting, which provides fast access-control list processing, accounting, and switching/routing with minimal impact on performance. One of the side benefits of enabling flow routing is that you can then "capture" a record of the flows as they are removed from the flow cache, creating what we call a "flow log." Among other things, the flow logs contain the source and destination IP addresses for all IP packets, source and destination port numbers for UDP and TCP traffic, and packet and octet counts.

Mark wrote the bulk of the OSU flow tools collection. Suresh wrote the flow-host-profile program. Steve mostly just uses the tools and makes suggestions for further development.

The University Technology Services Networking Group at The Ohio State University has written a suite of tools to record, filter, print, and analyze flow logs. Our initial interest was in computer intrusion response. When an intrusion is reported on campus, we can readily search the captured flow logs to determine when the initial attack occurred and from what IP source; where else the attacking host attempted to connect on our networks; and what network traffic ensued from the victim host after the intrusion. Although the flow logs do not contain packet-content information, the level of detail is sufficient to get a very detailed picture of network activity for our site.

We recognized early on that we should be able to use the flow logs for some types of intrusion detection, possibly in near real-time. This article describes what the flow logs contain, the tools we have written for intrusion detection, and how we use those tools. We also point to related references you might find interesting.

What the Flow Logs Are

We'll start with Cisco's description of flows:

> A network flow is defined as a unidirectional sequence of packets between given source and destination endpoints. Network flows are highly granular; flow endpoints are identified both by IP address as well as by transport layer application port numbers. NetFlow also utilizes the IP Protocol type, Type of Service (ToS) and the input interface identifier to uniquely identify flows.
>
> Non-NetFlow enabled switching handles incoming packets independently, with separate serial tasks for switching, security, services and traffic measurements applied to each packet. With NetFlow-enabled switching, security (ACL) processing is applied only to the first packet of a flow. Information from the first packet is used to build an entry in the NetFlow cache. Subsequent packets in the flow are handled via a single streamlined task that handles switching, services and data collection concurrently.
>
> (Source: *NetFlow Services and Applications*, <http://www.cisco.com/warp/public/cc/cisco/mkt/ios/netflow/tech/napps_wp.htm>)

Flow logs are a record of the flows created by NetFlow accounting. These logs contain interesting information about network traffic, including source and destination IP addresses and port numbers, the time for the beginning and end of the flow, count of packets and octets, and so on.

A flow whose TCP flag includes the FIN bit but not the SYN bit covered either packets from the middle and end of a TCP connection or packets from a "stealth" scanner.

It is probably easiest to explain by example. Suppose that I successfully telnet from host A port 1234 to host B port 23. The initial packet from A to B causes the router to create a flow entry for {TCP,A,1234,B,23}. The response from B to A causes the router to create a related flow {TCP,B,23,A,1234}. Data from subsequent traffic will be aggregated in these two flow records either until the TCP session terminates or until the flow records are removed from the cache, either because they have timed out or because the cache is full. The flow records contain a count of the number of packets and octets seen in that flow. For TCP traffic, the flow records also record the OR of all of the TCP header-flag bits seen in the flow so far. Flow records also record the IP protocol (TCP, UDP, ICMP), and router interfaces that the traffic was received on and sent to.

One can see from this example that most network interactions will result in at least two flows, one for traffic going in each direction. It is important to note that a TCP session may consist of far more than two flows. You can generally tell which flows "belong together" by matching the IP addresses and port numbers, and you can tell whether a flow "contained" the packets for the start, middle, or end of the connection by looking at the TCP-flag bits. For example, a flow whose TCP flag includes the FIN bit but not the SYN bit covered either packets from the middle and end of a TCP connection or packets from a "stealth" scanner.

Flows for UDP and ICMP traffic are similar, although it is important to note that since neither of these is a connection-oriented protocol, flows of UDP and ICMP traffic are just collections of "similar" packets.

Flow Tools

Ohio State has written a suite of tools for collecting, filtering, printing, and analyzing Cisco flows. The tools are written to work as UNIX pipelined commands, making it easy to perform data reduction without creating unnecessary intermediate files.

We capture the flows on several Pentium-class computers running FreeBSD using a program called flow-capture. The capture hosts have enough disk space to hold a few days' worth of flow logs each. Flow-capture was designed specifically to handle the collection of high volumes of flow logs from multiple routers – our busier routers generate about 650MB per day of compressed log information. The captured logs are copied periodically to a 250GB RAID on a "decked out" host where we do most of our analysis of the flow logs. We retain a window of past logs to facilitate incident response and in anticipation of handling bill-dispute resolution once we move toward usage-based charges for billing.

We store flow records in snapshots that cover small intervals of time (currently 15 minutes). This way, the individual files are of a more manageable size, and we can limit our analysis and review efforts more specifically to time periods that we are interested in, rather than wade through large amounts of data we would otherwise have to skip through. The flow-cat program concatenates separate flow log files that are named on its command line into stdout. (We can't simply use the UNIX cat command, since each flow log file starts with a short header that has to be stripped out.)

The typical tools that we use for incident response are flow-filter, flow-print, and flow-search. Flow-print prints flow records in any of several forms. Flow-filter filters flow records using Cisco-style access control lists, allowing us to include or exclude records to limit what we are viewing. Flow-search is a script that makes it easier to apply flow-filter to a set of flow log files. It provides a simple command-line interface that allows you to filter on source or destination or to extract only traffic going between two sets of access control lists.

The toolkit also includes programs that perform summary and statistical analysis of the data in flow logs.

The two chief intrusion-detection tools are flow-dscan and flow-host-profile. Flow-dscan reads through a set of flow logs (e.g., for a 24-hour period) and reports on host and port scans, various sorts of floods, and a few other types of activity typically associated with intrusions.

start time	src ip	src port	dst ip	dst port	p	f	#	octets
00:00:11.380	164.107.1.2	1026	205.188.254.195	4000	17	0	1	56
00:00:11.384	216.65.138.227	1055	164.107.1.3	28001	17	0	1	36
00:00:11.384	164.107.1.3	28001	216.65.138.227	1055	17	0	1	68
00:00:11.392	164.107.1.4	27015	24.93.115.123	1493	17	0	3	1129
00:00:11.392	164.107.1.5	1034	205.188.254.207	4000	17	0	1	48
00:00:11.392	128.146.1.7	53	206.152.182.1	53	17	0	1	61
00:00:11.404	204.202.129.230	80	140.254.1.6	1201	6	3	30	14719

Figure 1. Sample output from flow-print, edited to make it more compact by deleting some columns, and changing some addresses to protect the guilty.

We can investigate each of these more thoroughly by using flow-search and flow-print to extract just the records pertaining to those hosts to see both what they were doing and whether it was really a scan or not, and also to see whether there was other activity that we should investigate more closely.

Flow-host-profile builds a list of the network services running on each host on campus and allows us to compare activity over a period of time with the existing profile to detect changes. We are especially interested in new hosts and new services that show up on campus. The sample output from flow-host-profile shows activity for several services that we had not previously seen (e.g., http services on 128.146.1.4). The new activity on port 7440 on 128.146.1.3 might be a new service, but sometimes busy clients have high-end port numbers that show up in the report. (As they get reused, the connection count goes up, passing our activity threshold.) We've been experimenting not just with looking at the presence of new services (and hosts), but also with changes in the level of activity of a service. For example, it would be interesting to know if activity on an usually quiet FTP server suddenly increases (possibly as a result of WAREZ trading through a writable directory, for instance).

port scan:	src=24.95.33.99	dst=164.107.3.40	start=924481109
host scan:	ip=209.252.199.29		start=924481109
host scan:	ip=209.249.159.31		start=924481109
host scan:	ip=205.188.3.177		start=924481109
host scan:	ip=209.249.159.59		start=924481109

Figure 2. (Edited) output from flow-dscan.

The detectors are not 100 percent foolproof, but flow-host-profile does report useful information that helps us keep abreast of computer security incidents involving our campus.

IP-ADDRESS	PORT	SERVICE	PROTO	CONNECTIONS	DAYS	PERCENT
128.146.1.1	53	domain	17	3	1	100
128.146.1.2	162	snmptrap	17	3	1	3
128.146.1.3	7440	N/A	6	3	1	100
128.146.1.4	80	http	6	3	1	100
128.146.1.5	518	ntalk	17	3	1	0

Figure 3. (Edited) output from flow-host-profile, showing new hosts and services; the PERCENT represents the change in usage pattern (or appearance of a new host/service) compared to historical data.

The OSU flow tools are available by anonymous ftp from <ftp.net.ohio-state.edu> in </users/maf/cisco/flow-tools.tar.gz>.

Summary

We have only limited experience with using the flow logs for intrusion detection. So far, the principal problem is false positives. Successful use of both flow-dscan and flow-host-profile depends on building and maintaining an accurate exclusion list to help eliminate false positives. Unfortunately, this would also tend to mask certain types of activity from being reported. The vast amount of data that we collect is useful, on the one hand, since we can "zero in" on network activity for debugging or investigation purposes, but it also presents certain challenges for designing accurate intrusion-detection systems, since one has to make a trade-off among accuracy of reporting, the level of analysis performed, and the system resources (especially memory) required for the analysis. We are exploring the use of better windowing to limit resource consumption.

The OSU flow tools are available by anonymous ftp from <ftp.net.ohio-state.edu> in </users/maf/cisco/flow-tools.tar.gz>.

Other groups have also been working on tools for collecting Cisco flow logs. In particular, you might be interested in looking at the CAIDA tools at <http://www.caida.org>. These seem to be oriented more toward statistical analysis of NetFlow logs, rather than incident response or intrusion detection. Cisco has also released a set of tools for collecting NetFlow logs that you can find at <http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc>.

invisible intruders: rootkits in practice

To catch a cracker you must understand the tools and techniques he will use to try to defeat you. A system cracker's first goal is to hide his presence from you, the administrator. One of the most widely used cracker tools for doing this is the rootkit. A rootkit gets its name not because the toolbox is composed of tools to crack root, but instead because it comprises *tools* to keep root.

by David Brumley

David Brumley works for the Stanford University Network Security Team (SUNSeT). He graduated with honors from the University of Northern Colorado in mathematics with additional work in philosophy. In his free time he enjoys playing hockey and reading Kant.

<dbrumley@stanford.edu>

Rootkits are used by intruders to hide and secure their presence on your system. An intruder achieves complete cloaking capability by relying on an administrator to trust the output of various system programs. This assumption is more or less true – most of the time system administrators trust ps to display all processes and ls to list all files.

The cracker hides simply by modifying these programs not to display his activities: ls is altered not to display the cracker's files, and ps is modified not to display the cracker's processes. This simple method proves powerfully effective. A system administrator often has no clue that anything is amiss. Should the administrator sense that the system does not "feel" right, she'll have a hard time tracking down the exact problem.

To replace any of the programs mentioned here, the cracker must already have root access. The initial attack that leads to superuser access is often very noisy. Almost every exploit will produce a lot of network traffic and/or log activity. Once in, though, the skilled attacker has no difficulty covering tracks. The average cracker will have programs in his rootkit such as z2 and wted that remove login entries from the wtmp, utmp, and lastlog files. Other shell scripts may clean up log entries in /var/log and /var/adm. Luckily, the average cracker is sloppy. Sometimes he will forget to clean out certain programs or will simply just zero out the log file. Any time a log file has zero length it should be an immediate sign that something is amiss.

Trojans

Once the cracker cleans up the appropriate files to hide his tracks, he will want to leave a backdoor in order to avoid using his noisy exploit again. Rootkit backdoors – often called trojan horses – can typically be divided into two categories: local programs and network services. These trojaned programs are the core of the rootkit.

Local programs that are trojaned often include chfn, chsh, login, and passwd. In each case, if the magic rootkit password is entered in the appropriate place, a root shell is spawned. Of course a smart cracker will also disable the history mechanism in the root shell.

The replacement for login is especially interesting. Since some systems have shadowed and unshadowed password schemes, the cracker's replacement must be of the right type. A careless cracker might use the wrong kind of login trojan. When this happens, all or some accounts will be inaccessible, which should be an immediate tipoff that a cracker has gained control of your system.

inetd, the network super daemon, is also often trojaned. The daemon will listen on an unusual port (rfe, port 5002 by default in Rootkit IV for Linux). If the correct password is given after connection, a root shell is spawned and bound to the port. The manner in which the shell is bound makes it essential to end all commands with a semi-colon (";") in order to execute any command line.

rshd is similarly trojaned. A root shell is spawned when the rootkit password is given as the username (i.e., `rsh [hostname] -l [rootkit password]` will get you in to the compromised machine).

Utilities Included in Rootkit IV

Programs That Hide the Cracker's Presence

ls, find, du – will not display or count the cracker's files.

ps, top, pidof – will not display the cracker's processes.

netstat – will not display the attacker's traffic, usually used to hide daemons such as eggdrop, bindshell, or bnc.

killall – will not kill the attacker's processes.

ifconfig – will not display the PROMISC flag when sniffer is running.

crontab – will hide the cracker's crontab entry. The hidden crontab entry is in /dev by default.

tcpd – will not log connections listed in the configuration file.

syslogd - similar to tcpd.

Trojaned Programs That Have Backdoors

chfn – root shell if rootkit password is entered in as new full name.

chsh – root shell if rootkit password is entered as new shell.

passwd – root shell if rootkit password is entered as current password.

login – will allow the cracker to log in under any username with the rootkit password. If root logins are refused, user rewt will work. It also disables history logging.

Trojaned Network Daemons

inetd – root shell listening on port rfe (5002). After connection, the rootkit password must be entered in as the first line.

rshd – trojaned so that if the username is the rootkit password, a root shell is bound to the port (i.e. rsh [hostname] -l [rootkit password]).

Cracker Utilities

fix – installs a trojaned program (e.g., ls) with the same timestamp and checksum information.

linsniffer – a network sniffer for Linux.

sniffchk – checks to make sure a sniffer is still running.

wted – wtmp editor. You can modify the wtmp.

z2 – erases entries from wtmp/utmp/lastlog.

bindshell – binds a root shell to a port (port 31337 by default).

Last, a root shell is often simply left bound to a port by the program bindshell. This program requires no password. By default the program is bound to port 31337, "eleet" in cracker jargon.

Satori

In all of these programs, the default password for the newest Linux rootkit (Rootkit IV) is `satori`. Older rootkits have used `lrkr0x` and `h0tb0x` as passwords. Rarely is the default left unchanged, but it never hurts to check.

To expand their domain, the cracker may also install an Ethernet sniffer. An Ethernet sniffer listens in on all traffic on the local network, grabbing passwords and usernames destined for other machines. ifconfig will normally report such odd behavior by alerting the administrator with the PROMISC flag. Unfortunately, ifconfig is usually one of the programs modified.

The allure of rootkits should now be obvious. Even if the administrator patches the program that initially led to root access, the cracker merely has to telnet to the proper port to get a root shell. If this is closed, the cracker can try the backdoored login or rshd program. And even if that doesn't work, the cracker can still log in as a user (from perhaps a cracked password or his Ethernet sniffer) and used the trojaned ping, chfn, or chsh program to become the superuser once again.

Why do crackers break into systems? Sometimes you are targeted directly. The cracker wants information or access specifically available at your installation. Often, however, a cracker may simply want to break into any system in order to get on IRC, serve up WAREZ, or trade MP3s. If they do this, they might trojan crontab in order to hide jobs that rotate, modify, or check on the status of the illicit activity.

Hidden

What tools does the administrator have to find these trojan-horse programs? If a rootkit is properly installed, the administrator will not be able to tell the difference between the original and a modified program. A widely used cracker program named fix will take a snapshot of the system binary to be replaced. When the trojaned or modified binary is moved into place, the fix program mimics all three timestamps (atime, ctime, and mtime) and CRC checksum of the original program. A carefully constructed and compiled binary will also have the same length.

Without a cryptographically secure signature of every system binary, an administrator cannot trust that she has found the entire rootkit. If even one program goes undetected, the intruder might have a way back into your system. Several utilities, such as tripwire and RedHat's rpm, provide secure MD5 checksums of binaries. To be truly secure, the reports must be kept offline in some sort of secure location, lest the hacker tamper with the report. (Not so long ago a system-cracker magazine called Phrack published an article on defeating online tripwire reports.) These reports may be the only thing that saves you from a complete reinstallation of the entire system.

Luckily, many crackers are careless, and portions of their rootkit can be detected. The trojaned files above often have configuration files that list the programs to hide and which to display. Often they forget to hide the configuration files themselves. Since /dev is the default location for many of these configuration files, looking in there for anything that is not a normal file is often a good idea. The default setup for many rootkits is to have the configuration file begin with `pty`, such as `/dev/ptys` or `/dev/pryr`.

Another trick is to look at modification times of all programs. Although a good cracker will try to cover most of the times, they often forget a few files or directories.

`find / -mtime -N -print`, where N is the number of days you expect the intruder has had access to your system, should work in most cases. I've found many times the hacker has covered his tracks well in /bin and /sbin, but left the entire build directory for his rootkit in /tmp!

Inside each modified directory you should compare the output of `echo *` with `ls`. If `ls` has been trojaned and configured to hide anything, the `echo` command will show it.

Also pay close attention to the strings in the system binaries. Although /sbin/inetd may look the right size, if the string "/bin/bash" shows up in it, you should start worrying about what else has been replaced. Another trick is to look at the file type. If file /bin/inetd says that inetd is not stripped, it most certainly has been tampered with.

If you're lucky enough to have a /proc filesystem, spend some time to become acquainted with it – there is a lot of useful information there. By walking the directory tree you can find which processes are running. After comparing the output to what ps shows, you can determine with some level of certainty whether ps has been modified. Other files in /proc may show you all active network connections, and some others may even list all open file descriptors!

The easiest way to detect crackers, however, is to have a clean set of statically linked binaries for your system. Statically linked? Sometimes a more advanced cracker will replace system libraries, so anything that dynamically uses them cannot be trusted. If possible you should have a spare set of common programs such as ps, ls, ifconfig, perhaps lsof, etc., on a secure host. When you find a compromised system, simply download the clean binaries, set your PATH environment variable to use them, and start looking for backdoors.

Various versions of rootkit are available at most cracker sites. The most accessible versions are for open-source operating systems such as Linux and FreeBSD. Also commonly reported are versions for Irix, SunOS, and Solaris. The latest rootkit, Linux Rootkit IV, is distributed by The Crackers Layer, <http://www.lordsomer.com>. It is definitely worth the bandwidth to download the source and see how it works.

Rootkits have become very popular tools for both experienced and novice crackers. Your first line of defense should always be protection with regular patches and administration. Equally important is the second line: a good plan in the event of a real compromise. By arming yourself ahead of time with secure checksums and clean binaries, you will be much quicker and more effective in local and sitewide incident response.

The easiest way to detect crackers, however, is to have a clean set of statically linked binaries for your system.

indicators of unix host compromise

What Intruders May Do After They Have Root

by Paul C. Brutch,

Paul C. Brutch, a 1999 USENIX scholar and Ph.D. candidate in computer science at Texas A&M University, performs research in network security.

<paulb@cs.tamu.edu>

Tasneem G. Brutch

Tasneem G. Brutch is a Ph.D. candidate in computer engineering at Texas A&M University. She performs research in security for wireless and mobile computing.

<tasneem@cs.tamu.edu>

and Udo Pooch

Dr. Udo Pooch is the E-Systems Professor of Computer Science at Texas A&M University.

<pooch@cs.tamu.edu>

Chronology of a Host Compromise

At 12:00 noon on July 19, 1995, a team of intruders decided to attack an organization's computer system – a system on which the intruders already had valid user accounts. Since they knew the organization's personnel, they decided to use social engineering. Their goal was to obtain the root password for the organization's primary server, and from there to wreak havoc. At 1:00 pm, using administrative privilege at another site, the intruders created a user account at that site with the same account and user name as those of the system manager of the target organization. The intruders sent an email message from this fake account to one of the system administrators in the target organization, asking for the server root password. At 3:11 pm, the system administrator replied to the spoofed email message and provided the real root password. The system administrator, replying out of elm, did not see the mail header and was therefore unaware that this message did not originate from the system manager.

At 3:12 pm, the intruders logged in to the primary server of the target organization using the account of another system administrator, whose password they had sniffed earlier. Next they `su`'d to root using the password sent to them in the email reply. They proceeded to change the root password and the password of the system administrator's account that they had used to log in. Next they removed other user accounts and even changed the EEPROM password using the `eeprom` command. Finally, the intruders attacked other machines on the organization's network and destroyed security and administrative functions.

Sometime before 4 pm, the system-administration team detected a problem: they could not log in. They noticed a message on the console of a client machine indicating that an unprivileged user had successfully su'd to root. They also found that the EEPROM password was changed on the server. At 4:12 pm, they decided to power off their machines. On the next day, July 20, the system-administration team struggled unsuccessfully to recover. By July 21, they decided that it was necessary to perform a complete reinstallation of all Sun workstations with an upgrade from SunOS to Solaris 2.4. On July 24, they had their machines operational, and user accounts were restored on July 25. On July 31, one of the client machines was compromised again by the same team of intruders.

This is a description of an actual incident. The individuals who performed this attack were later identified. Instead of being prosecuted, they were rewarded. The attack was accomplished by graduate students during a computer-security laboratory exercise conducted at Texas A&M University. The attack was real, but it was made in a controlled environment as part of a graduate-level computer-security course. This article focuses on some of the ways that students in the course were able to compromise a UNIX host and what actions they took to hide their presence and maintain superuser privilege.

Overview of the Course

Dr. Udo Pooch started the graduate computer-security course in the summer of 1995; through the spring 1999 semester, he has taught it to over 110 students. The course is a mixture of formal classroom instruction on computer and network security principles, and a hands-on security laboratory. As part of the security laboratory, students are divided into multiple penetration teams and a single system-administration team. Each penetration team is given superuser access to a Linux machine residing on a private network. A penetration team has complete control over its assigned Linux machine, and the system-administration team is not normally allowed to venture onto the penetration team's network. The system-administration team manages machines on a separate network, and these two networks are connected by a router. The system-administration team's network consists of a number of Sun workstations running Solaris 2.5.1 and one NT 4.0 machine. The router isolates the security laboratory from the department's network, and a proxy server provides a controlled single point of access to the laboratory[4].

In some cases, penetration teams have compromised a monitored host before the system-administration team was even able to install all of its security tools.

The goal of the penetration teams is to compromise a machine managed and monitored by the system-administration team. The penetration teams are allowed to make almost any type of attack as long as their activity remains within the domain of the security laboratory. The penetration teams have accounts on their own Linux machines and separate user accounts on some of the system-administration team's machines. Therefore, the penetration teams can conduct attacks as inside intruders and simulate remote attacks from the Internet. The system-administration team also provides one Sun workstation running Solaris 2.5.1, without any security patches, for use as a training machine by the penetration teams. Although this training machine resides on the system-administration team's network, it is not trusted by any of the other machines and it is not monitored by the system-administration team. Penetration teams have successfully launched attacks from this training machine to compromise more secure hosts on the system-administration team's network.

The goal of the system-administration team is to detect and trace all unauthorized access for the machines that it manages and monitors. The system-administration team makes every effort to ensure that the systems they monitor are secure. Ideally, the system-administration team should: install the latest vendor security patches; perform vulnerability scanning by running Tiger scripts by Doug Schales; install tcp wrapper by Wieste Venema to monitor and filter incoming requests for certain network services; run Crack by Alec Muffet against the password file; enable remote logging via the syslog facility; and run Tripwire by Gene Kim and Eugene Spafford to perform system integrity checking. Unfortunately, the system-administration team spends much of its time at the beginning of each semester performing mundane administrative tasks like setting up user accounts. In some cases, penetration teams have compromised a monitored host before the system-administration team was even able to install all of its security tools.

Throughout the past five years, various hardware and software configurations have been installed in the security laboratory. For example, in 1998 secure hubs were used for physical connectivity to prevent penetration teams from sniffing traffic on the system-administration team's network[4]. The security laboratory changes each year as new system-administration teams try different configurations to implement different security solutions. As the security laboratory configuration becomes more complex, it requires more time from the system-administration team to set up and manage.

A symbolic-link attack was made against the Solaris Admintool, which uses an insecure locking mechanism to allow simultaneous access, modification, and creation of files by various users.

Definition of Host Compromise

We need to define what we mean by a host compromise. Kahn states that a component is compromised if it is misused or under the control of an adversary of its owner; events that can compromise a component include penetration, attack by an insider, and accidental misuse[3]. For the purpose of this article, we will say that a UNIX host is compromised whenever an intruder is able to log in with the real user ID and effective user ID of superuser, or when an intruder can change the real user ID or effective user ID of a process, such as a shell program, to superuser. In UNIX, any account with UID 0 has superuser privilege.

At login time, intruders can set their real and effective user ID to superuser simply by logging in as the root account and supplying the root password. Any other account that has a UID of 0 would work. Intruders sometimes create or modify accounts in the `/etc/passwd` file with UID 0 as a backdoor. Alternatively, the intruder can use the `setuid()` routine to set the real and effective user ID of a process. In order for this attack to work, the process calling the `setuid()` routine needs to have an effective user ID of superuser. Root-owned, set-user-ID files (programs) temporarily change the effective user ID of the process to superuser when the program is executed. In some cases, it is necessary to provide root-owned, set-user-ID programs in order to allow normal users to perform tasks that require privilege. Intruders often exploit these set-user-ID programs to run a process, such as a shell program, with an effective user ID of superuser.

Exploits to Gain Superuser Privilege

The penetration-team members used several approaches to acquire superuser access on the system-administration machines. These included exploitation of system misconfigurations, buffer-overflow attacks, symbolic-link attacks, TCP spoofing, and social engineering. One of the penetration teams took advantage of the trusted status of their machine in the `/etc/hosts.equiv` file on one of the system-administration team machines to rlogin as a privileged user. Buffer-overflow attacks were made against root-owned, set-user-ID programs such as `fdformat` and `eject`. These programs had vulnerabilities because bounds checking was not properly done in the volume-management library. Symbolic-link attacks were also run against root-owned programs such as `lpr` and Solaris Admintool. The penetration teams used repeated invocation of the lpr command to overwrite the `/etc/passwd` file using a previously created symbolic link in the print spool directory. Another symbolic-link attack was made against the Solaris Admintool, which uses an insecure locking mechanism to allow simultaneous access, modification, and creation of files by various users. This vulnerability was exploited to create world-writable files that were owned by root.

In one of the laboratory configurations, source routing was left enabled by the system-administration team. This allowed a penetration team to launch a successful TCP spoofing attack. The attack was launched from the training machine, which was compromised by a penetration team, on the system-adminitration team's network. The attack exploited the fact that machines on the network, with the exception of the training machine, trusted each other.

From the training machine, the penetration team made a SYN flood attack against another system-administration team's machine. Then the penetration team, using the training machine, was able to impersonate the attacked machine and exploit its trusted relationship with other machines in the network. The penetration team used snoop to acquire TCP sequence numbers, and then as user bin established a connection with

rshd on a target machine in the system-administration team's network. As user bin, they sent spoofed packet(s) containing the command `cp /bin/sh /tmp/sh2; chmod 4755 /tmp/sh2` to the target machine. Next the penetration team found that the root crontab file on the target machine was world-readable and had an entry to run a bin-owned program. They modified this called program to create a root-owned, set-user-ID shell. Last but not least was the penetration team that used social engineering to acquire the root password – the incident we described at the beginning of this article.

Covering Their Tracks

Once a penetration team successfully exploited a vulnerability to acquire superuser privilege, it attempted to hide this activity from the system-administration team. Penetration teams normally changed the UNIX system log files, security tools, and the access and modification times of modified system files to cover their tracks. Some penetration teams, however, did not care if their attacks were detected. Instead, they concentrated on ensuring that the attack could not be traced back to their account. These penetration teams made their attacks by masquerading as other legitimate users.

Penetration teams used superuser privilege to remove their presence from the UNIX system log files. In particular, they removed their entries from wtmp, wtmpx, utmp, utmpx, and lastlog using a number of log-editor programs. Two such log-editor programs are UTClean.c and marry.c, which are available from <http://www.rootshell.com>. Other log files that were edited by the penetration teams were messages and sulog. Some teams locked the system log files using the command `/usr/bin/chmod 000` at the beginning of every attack. Other teams linked system log files to /dev/null. Even worse, some penetration teams simply removed the log files or the entire /var directory. This action usually caused severe problems for the attacked machine.

Some penetration teams simply removed the log files or the entire /var directory. This action usually caused severe problems for the attacked machine.

In some configurations, the syslog facility was used to provide remote logging to a secure host. One penetration team used an attack to kill syslogd. This attack exploited a vulnerability of Solaris 2.5.1 – that syslogd will crash if it receives an external message from a sender with an unresolvable IP address. The details of this attack can be found in sol_syslog.txt, which is available from <http://www.rootshell.com>.

In addition to modifying the UNIX system log files, penetration teams also targeted the security tools installed by the system-administration team. One of the security tools modified was Tiger scripts by Doug Schales. A penetration team modified the system-administration team's copy of Tiger scripts so that it ignored the penetration team's root-owned, set-user-ID shells. Such attacks were successful because the system-administration teams left their security tools on the compromised host.

In order to hide changes they made to any system file, penetration teams modified the access and modification time of a file using the touch utility. The inode change time of a file was set to the time that the touch utility was used. The system-administration teams were, however, able to view the inode change time using the command `ls -cl`.

The penetration teams masqueraded as other users by stealing their passwords. The methods used to acquire user passwords included password cracking, password sniffing, X-Window spying, and Trojan-horse login programs. The students in penetration teams ran password-cracking tools against the world-readable /etc/passwd file. Some of the tools used were Crack by Alec Muffet and John the Ripper by Solar Designer. Penetration teams were also successful in sniffing passwords when they had superuser access on the same network as the system-administration teams machines and secure hubs were not used. The sniffers used by the penetration teams included tcpdump by Van Jacobson, snoop (which is packaged with Solaris), and sniffit by Brecht Claerhout.

The most common backdoor was a root-owned, set-user-ID copy of /bin/sh.

Some of the penetration teams wrote their own filters to analyze the data captured by sniffers. The program xkey, by Dominic Giampaolo, was used by one of the penetration teams to spy on users' sessions by capturing their keystrokes on an X server, with access control disabled. Some penetration team members used Trojan-horse login programs for /bin/login, /bin/rsh, and /bin/rlogin to capture user login IDs and passwords.

Leaving a Backdoor

In case the system-administration team detected and fixed the exploited vulnerability, the penetration teams left an alternate backdoor to gain superuser privilege on the system. The most common method was to leave a root-owned, set-user-ID shell as a hidden file or to modify the /etc/passwd file. Other backdoors included Trojan-horse programs and modified network services. Penetration teams also modified the boot startup scripts and cron schedules to create set-user-ID shells.

The most common backdoor was a root-owned, set-user-ID copy of /bin/sh. These shell programs were most often left in the penetration team's home directories, the /tmp directory, and scattered throughout the system as hidden files. The next most common backdoor was the modification of the /etc/passwd file. Penetration teams created user accounts with a user ID (UID) 0, which provided the account superuser privilege. A number of other modifications were made to the /etc/passwd file, including creating accounts without passwords, adding a password to the smtp account, and changing the root password.

A number of Trojan-horse programs, included Trojan login and su, were used as backdoors. Modified copies of login and su were installed in /usr/bin to grant superuser access to a predefined password. Rootkits usually provide these types of Trojan programs.

The network services were also a major target for the penetration teams. The tcp wrapper program by Wieste Venema provides access control of incoming network service requests for certain services using the hosts.allow and hosts.deny files. These files were modified by the penetration teams to allow certain network service requests that were denied by the system-administration team. Another team created its own network server, which started a root-owned copy of /bin/sh. This penetration team modified the /etc/inetd.conf file and /etc/services to add this network service to the system. They also added false comments and fake documentation for this service in order to make it appear legitimate. Another penetration team installed a fake lpsched daemon that was actually a remote procedure call (RPC) server running in the background as root, waiting to accept commands from a remote RPC client. The RPC server would accept the commands chmod `4755 /usr/bin/ksh` or `chmod 555 /usr/bin/ksh` from the RPC client.

In case the system-administration team were to discover and remove their backdoors, some penetration teams left an alternate mode to gain superuser access. They modified the boot startup scripts and the cron schedule to create root-owned, set-user-ID shells. One team, for example, created a script called S21fsflush in the /etc/rc2.d/ directory. Other teams modified the root crontab file.

For More Information

If you are looking for more details on these attacks, a survey paper on the penetration tests performed during the 1995, 1997, and 1998 security classes that was presented at the SANS Network Security '98 Conference is available in the conference proceedings[2]. A version of the survey paper is also available online as a technical report, TR

98-021, from the Department of Computer Science at Texas A&M University.

If you are interested in starting your own laboratory to perform security vulnerability testing and analysis, we recommend that you read Marti, Bourne, and Fish's paper, "CPSC 665 Advanced Networking and Security Game Administration Plan"[4] and Bishop and Heberlein's paper, "An Isolated Network for Research"[1].

Acknowledgments

We would like to acknowledge the following students whose reports were analyzed for the compilation of this article:

1995 Class: B. Devalla, E. Mitchell, C. Schroeder, S. Sundaresan, R. Wood, P. Brutch, J. Jansen, L. Crotwell, D. Nash, Shiqian Yu, M. Woodings, Alex.

1997 Class: J. Bourne, D. Derrick, B. Fish, D. Ragsdale, S. Joshi, B. Goteti, S. Veluswamy, A. Madhwaraj, M. Sonaseth, D. Lazarova, Y. Feng, M. Guo, Q. Shi, W. Son, R. Agni, Sudhindra N., E. Khan, K. Kothandaraman, C. Standish, T. Brutch, P. Brutch, W. Marti.

1998 Class: X. Ma, L. Zheng, J. Zhao, W. Feng, H. Kim, M. Kulvicki, R. Mixer, R. Kaimal, S. Ramachandran, P. Dakshinamoorthy, S. Dhanekula, Y. Yu, T. Wang, Q. Zhou, J. Bourne, B. Fish, M. Hines.

References

[1] Bishop, M., and Herberlein, L. "An Isolated Network for Research." The 19th National Information Systems Security Conference. 1996.

[2] Brutch, P.; Brutch, T.; Mitchell, E.; and Pooch, U. "UNIX Penetration Tests: Attempts Performed During a Graduate Security Class at Texas A&M." SANS Network Security 98, Technical Conference Part 1, October 24-31, 1998.

[3] Kahn, C. "Using Independent Corroboration to Achieve Compromise Tolerance." 1998 Information Survivability Workshop, October 28-30, 1998.

[4] Marti, W.; Bourne, J.; and Fish, B. "CPSC 665 Advanced Networking and Security Game Administration Plan." WECS '98, Workshop on Education in Computer Security, 19-21 January 1998.

a hacker's approach to id

by Mudge

Mudge is a consultant and programmer. His paper, co-written with Bruce Schneier, about vulnerabilities in PPTP resulted in a free dinner from Microsoft.

<mudge@L0pht.com>

This article is a response to Rik Farrow's question about the L0pht's work on intrusion-detection packages for Network Flight Recorder. In particular he asked how we chose which packets to look at. So I shall attempt to give a brief overview of how a group of hackers – and I use the term in the good sense – goes about approaching network intrusion detection, given the current state of tools and environments.

A little background first – but don't worry, I'll try to make it as painless as possible. For those not familiar with the L0pht, I recommend checking out the Web site, <http://www.L0pht.com>. (Note: that is L-Zero-p-h-t.) What we are, in a nutshell, resembles a marriage between Consumer Reports and public television . . . gone high-tech-security happy. One of the many products/tools/technologies that we played with happened to be Network Flight Recorder's tool of the same name. NFR (<http://www.nfr.net>) was designed to be a black-box recorder of packets going across a network. If you imagine an RMON probe on steroids you have NFR. Learning what types of traffic, who the heavy talkers on the net are, where different servers are located, and logging invalid packets and whether they come in the form of invalid checksums are all functions that NFR was designed to be capable of. Some astute readers might notice that nowhere have I mentioned that NFR was designed explicitly to be an intrusion-detection system – simply a versatile sniffer with an extensible programming language, called "N-Code," for handling packets. I had downloaded a copy from NFR's Web site and, along with fellow L0pht member Silicosis, whipped up a few simple N-Code modules that would handle some trivial intrusion scenarios and posted them on our Web site for everyone to access free of charge. NFR's president and CEO, Marcus Ranum, saw them and approached us to write a more complete set of filters (we are currently at over 200 checks and less than a third done) under contract for NFR. Since we had toyed around with the notion of writing them for free we jumped at the opportunity to put some food on the table. Gee, I hope Marcus does not read this magazine. Hi Marcus – d'oh!

A Deep Dark Secret

So now you have your crash course background on the L0pht and the tool NFR. Let's move on to current network intrusion-detection systems (IDSes). You have been exposed to all of the sales pitches, marketing literature, and preacher sermons on them. The CEO of the company wants to practice due diligence and comply with industry standards and deploy ID systems to catch all of those malicious crackers that Big Blue shows you in TV commercials daily. Now, as the curtain is slowly being withdrawn, I will tell you the deep dark secret about IDSes: they don't work.

What? Say it ain't so, Mudge! Unfortunately, it is indeed true that network ID systems do not work. At least not as advertised. Maybe it is the overzealous marketing and sales forces. Maybe it is the public's huge desire to have this mythical beast. Whatever it is, do not place all of the blame on the software. Sure, some are better than others (and often price has nothing to do with that) but, if you think about it, we are asking for a solution to an almost impossible problem. At least with today's technology and a realistic budget.

A Tall Order

It would be prudent for me to elaborate on my definition of a network intrusion-detection system. Then I can explain why I, and many other people, believe that the

current systems simply cannot work as well as the vendors would like you to believe.

My definition is:

> A device that passively monitors all of the traffic on a network, noticing and logging malicious patterns with extremely high accuracy and extremely low false positives. The device does not degrade network performance and will not miss detecting real attacks, cannot be tricked into logging incorrect data (false alarms), and cannot be disabled through a denial-of-service attack.

This might seem like a tall order to fill, but take a look at the next advertisement you see for one of these devices, or maybe even the recommendations from an auditing company on what is required for due diligence, and see if this is not alluded to.

Now, this is not to say that I do not feel that network IDS systems are valuable, just that they are nowhere near the panacea some would have you believe. I firmly believe that the proper use of these systems can help raise the sophistication level or difficulty required to compromise systems on your network. A huge component of proper use is understanding the systems' limitations.

The excellent paper by Tim Newsham and Thomas Ptacek, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" <http://www.nai.com/products/security/advisory/papers/ids-html/doc000.asp> is absolutely required reading for those wishing to explore the shortcomings of network IDSes.

In a nutshell, what one is asking of the system is for it to do the following:

- Never drop a packet.
- See all of the network traffic.
- Understand what the TCP window is for a particular system.
- Handle packets out of order.
- Handle all of the bizarre aspects of fragmented packets (short fragments, overlapping fragments).
- Correctly handle "duplicate" packets. (Which packet did the end system really process? Was the checksum correct? Does that matter?)
- Understand exactly how many hops away every destination is to prevent TTL attacks. (Did the end system even see the packet that the IDS just logged, or did it expire in transit?)
- Have infinite resources. (If state is being held for a session, what happens when it is kept open indefinitely? If there are 3,000 sessions open and the signature that one is attempting to match is extremely processor intensive, what happens?)
- Never "fail open" (which is impossible, since it is a passive device).
- Understand how applications handle data. (In a telnet session does the IDS correctly filter out telnet options, or does their inclusion cause the pattern matching to fail? If a client is in character mode, how does the end node handle backspaces – is `rb^Hoot` really the same as `root`?
- Understand IP options.
- Etc., etc.

A Bare Minimum

None of the network ID systems or environments that I am currently aware of do all of the above. Still, in a fit of insanity we decided that with only a few of the above requirements we could use some common sense and create what we felt was a *due-diligence-industry-standard solution.* Ewww – don't you just hate "market-ese"! Here is our list of

Two of our most rewarding common-sense approaches to optimizing and writing the filters are based on the following:

- *Model as many things as possible into state engines.*
- *Flag anomalies as according to RFC specs.*

requirements. Feel free to disagree with them, use them as your own, add to them, or whatever.

The system must:

- Have an extensible language to allow custom handling, analysis, and manipulation of packets. This must give direct access to any portion of the frame that is requested. In addition, the code must execute quickly, since all processing of packets in a stream is basically an inner loop.
- Be able to handle packets arriving out of order (nonsequentially) and order them for the IDS programmer.
- Not ignore fragmented packets.
- Have a notion of state for stateful sessions such as TCP.
- Be able to handle at least a 10MB Ethernet segment running at full bandwidth (we won't go into the fact that this requires only two systems talking, since heavily populated networks will peak out at roughly 33% due to collisions) without dropping packets).
- Perform valid checksum routines of packets witnessed.
- Run on a system that is capable of being secured and remotely managed in an encrypted fashion.
- Be extensible, programmable, extensible, programmable, ad infinitum.

I strongly believe that the above is a bare minimum that all customers need to demand from their network IDS vendors. If your current product does not do the above, beat on your vendor for them to add it. You are not only helping yourself but enabling your vendor to supply a better product.

Our choice ended up being NFR running on OpenBSD managed through SSH tunnels or SSL'd Web connections. One of the best things that we felt NFR had going for it, other than providing access to the source code (which was a big plus), was that the main engine seemed much more like an IP stack than some of the competitors at the time. This was most likely due to the product not initially being designed explicitly for IDS but as a programmable network monitoring and logging tool. All of this by no means implies that this is the only decent solution or even the best solution. There are far too many variables in people's needs for me to assume that. It fits some of our needs; maybe it will fit yours too – YMMV.

Two of our most rewarding common-sense approaches to optimizing and writing the filters are based on the following:

- Model as many things as possible into state engines.
- Flag anomalies as according to RFC specs.

By modeling sessions into state engines, we gained twice: increased performance and minimized false positives. Take sendmail or nntp as examples. Both have very similar state engines that can be overlaid on them. There are command, header, and data sections. Command sections would be VRFY, MAIL, HELO, POST, IHAVE, etc. Here is where you would look for the infamous 8.6.12 syslog overflow attacks, WIZ/DEBUG attacks (it is a shame that because of marketing, IDSes still spend cycles looking for this), mailing to programs, and on and on. One should not be wasting precious cycles looking for keyword matches that they expect to find in the data portion of a message.

Likewise, one should not flag a false positive every time the data content of some mail from a programming-language-related mailing list contains the word DEBUG.

```
[ command ]
data
[ header ]
<blank>
[ data section ]
.
[ command state again ]
```

By following the above flow between states, an optimized filter can be created that spends cycles looking for particular attacks only where they will actually be found.

With flagging anomalies according to RFC specifications, I am alluding largely to your friend and mine, the common buffer overflow. In addition to having a list of known buffer overflows and the signatures for them, it is prudent to look for new ones that you are not aware of yet. This is accomplished in many cases by watching for lengths that exceed protocol specifications. If a particular protocol states that in command state, no line will be more than 256 bytes including the command and whitespace – does it not seem wise to log the 4096-character-long IHAVE command that just went by? There are a surprising number of places where this works quite well.

So while we still have our doubts as to the future of network IDSes, we have no questions about how to get around them. But if you can drop the noise level down and catch the common script kids and crackers, that leaves more time to play the game with the true brilliant ones. Chances are you will learn a lot more this way – and isn't that so much more interesting?

If you can drop the noise level down and catch the common script kids and crackers, that leaves more time to play the game with the true brilliant ones.

a glimpse into the future of id

Cyberspace is a complex dimension of both enabling and inhibiting data flows in electronic data networks. Current-generation intrusion-detection systems (IDSes) are not technologically advanced enough to create the situational knowledge required to monitor and protect these networks effectively. Next-generation IDSes will fuse data, combining short-term sensor data with long-term knowledge databases to create cyberspace situational awareness. This article offers a glimpse into the foggy crystal ball of future ID systems.

by Tim Bass

Tim Bass is the CEO and managing director for Silk Road, a consulting business in Washington, D.C. specializing in network design, management, and security.

<bass@silkroad.com>

and Dave Gruber

Dave Gruber, Lt. Col., is the communications squadron commander at Hickam AFB, Hawaii.

<david.gruber@hickam.af.mil>

Before diving into the technical discussion, we ask the reader to keep in mind the generic model of a datagram traversing the Internet. Figure 1 illustrates an IP datagram moving in a store-and-forward environment from source to destination; it is routed on the basis of a destination address with an uncertain source address decrementing the datagram time-to-live (TTL) at every router hop[1]. The datagram is routed through major Internet and IP transit providers.

There is a striking similarity between the transit of a datagram on the Internet and an airplane through airspace, between future network management and air traffic control (ATC). At a very high abstract level, the concepts used to monitor objects in airspace apply to monitoring objects in networks. The Federal Aviation Administration (FAA) divides airspace management into two distinct entities. On the one hand, local controllers guide aircraft into and out of the airspace surrounding an airport. Their job is to maintain awareness of the location of all aircraft in their vicinity, ensure proper separation, identify threats to aircraft, and manage the overall safety of passengers. Functionally, this is similar to the role of network controllers, who must control the environment within their administrative domains. The network administrator must ensure that the proper ports are open and that the information is not delayed, that collisions are kept to a minimum, and that the integrity of the delivery systems is not compromised.

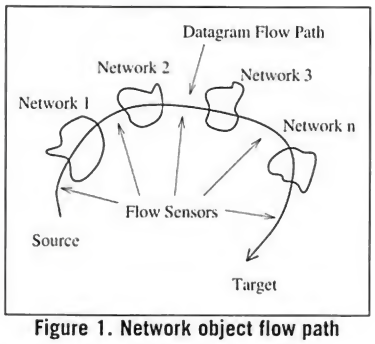


Figure 1. Network object flow path

This is similar to the situational awareness required in current-generation ATC. The FAA controls the routes between source and destination (airports), and airport authorities control the airports (as both router and host), maintaining the safety of the payload (passengers) and the transport agent (the airplane). The success of ATC depends on the fusion of data and information from short-term and long-term knowledge sources to create airspace situational awareness. This role is remarkably similar to network operators in future complex internetwork environments. As an example, consider the FAA and the National Weather Service as they monitor the weather. A change in environment can cause the FAA to make changes in air routes and landing criteria. This is similar to service providers keeping an eye out for unfavorable conditions in networks – for example, the loss of a major Internet transit network; severe congestion on major interdomain links; or attacks against routers, computers, and information. The same data-fusion concepts are shared across the airspace management functions and organizations. We expect that a similar fusion paradigm will occur with network management, Internet Traffic Control (ITC), and future intrusion-detection systems. Of course, this will not occur overnight (and may never become as comprehensive as ATC), but the analogy does help provide a glimpse into the future of ID.

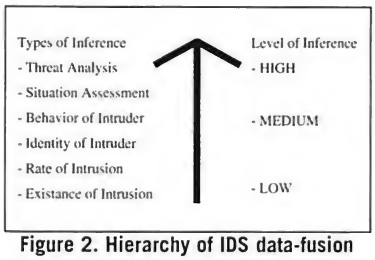


Figure 2. Hierarchy of IDS data-fusion inferences

Figure 2 illustrates the levels of situational knowledge inference required to support both the air traffic controller and the network manager. Sophisticated electronics must

identify objects against a noise-saturated environment, track the objects, calculate their velocity, and estimate the projected threat. These are nontrivial technical requirements.

Experienced network-security professionals generally agree that current-generation intrusion-detection systems are not technically advanced enough to detect multiple, complex non-signature-based cyberattacks, illustrated in Figure 3. Next-generation cyberspace IDSes require the fusion of data from heterogeneous distributed network sensors, modeled in Figure 4.

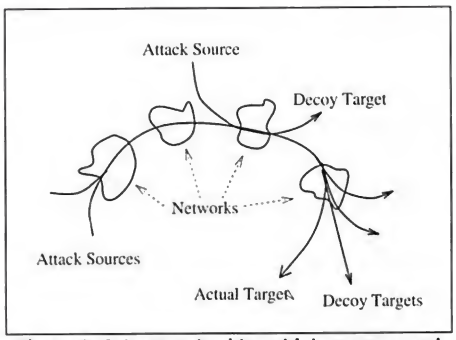


Figure 3. Cyberattack with multiple sources and targets

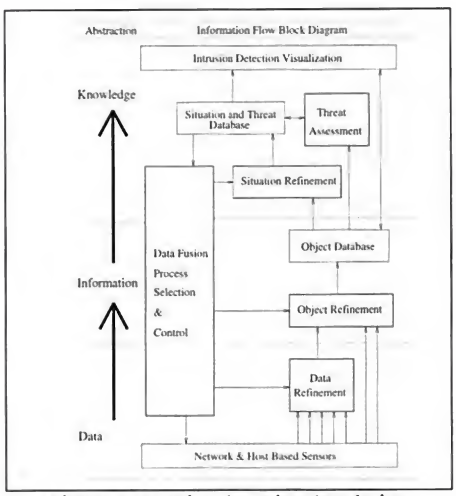


Figure 4. Intrusion-detection data fusion

Historical Intrusion Detection Systems

We offer a brief review of the state of the art of current-generation ID systems, from our recent ACM paper[2].

Internet ID systems historically examine operating-system audit trails and Internet traffic[5, 6] to help insure the availability, confidentiality, and integrity of critical information infrastructures. ID systems attempt to protect information infrastructures against denial-of-service attacks, unauthorized disclosure of information, and the modification or destruction of data. The automated detection and immediate reporting of these events are required to respond to information attacks against networks and computers. The basic approaches to intrusion detection today may be summarized as: known pattern templates, threatening behavior templates, traffic analysis, statistical-anomaly detection, and state-based detection. These systems have not matured to a level where sophisticated network-centric attacks are reliably detected, verified, and assessed.[2]

Computer intrusion-detection systems were introduced in the mid-1980s to complement conventional approaches to computer security. IDS designers often cite Denning's[5] 1987 intrusion-detection model built on host-based subject profiles, systems objects, audit logs, anomaly records, and activity rules. The underlying ID construct is a rules-based pattern-matching system whereby audit trails are matched against subject profiles to detect computer misuse based on logins, program executions, and file access.

The subject-anomaly model was applied in the design of many host-based IDSes, among them Intrusion Detection Expert System (IDES)[7]; Network Intrusion Detection Expert System (NDIX)[9]; and Wisdom & Sense (W&S), Haystack, and Network Anomaly Detection and Intrusion Reporter (NADIR) [10]. Other ID systems are also based on the Denning model; an excellent survey of them may be found in Mukherjee et al.[6]. The basic detection algorithms used in these systems include:

- weighted functions to detect deviations from normal usage patterns
- covariance-matrix–based approaches for normal usage profiling
- rules-based expert-systems approach to detect security events

The second-leading technical approach to present-day intrusion detection is the multi-host network-based IDS. Heberlein et al. extended the Denning model to traffic analysis on Ethernet-based networks with the Network Security Monitor (NSM) framework[11]. This was further extended with the Distributed Intrusion Detection System (DIDS), which combined host-based intrusion detection with network-traffic monitoring[6, 8]. Current commercial IDSes such as Real Secure by ISS and Computer Misuse Detection System (CMDS) by SAIC have distributed architectures using either rules-based detection, statistical-anomaly detection, or both.

A significant challenge remains for IDS designers to fuse sensor, threat, and situational information from numerous heterogeneous distributed agents, system managers, and

The output of fusion-based ID systems consists of estimates of the identity (and possibly the location) of a threat source, the malicious activity, taxonomy of the threats, the attack rates, and an assessment of the potential severity of damage to the projected target(s).

databases. Coherent pictures that can be used by network controllers to visualize and evaluate the security of cyberspace is required. Next, we review the basic principles of the art and science of multisensor data fusion applied to future ID systems in Bass[2] and Bass[3] to create highly reliable next-generation intrusion-detection systems that identify, track, and assess complex threat situations.

Internet Situational Data Fusion

In a typical military command-and-control (C2) system, data-fusion sensors are used to observe electromagnetic radiation, acoustic and thermal energy, nuclear particles, infrared radiation, noise, and other signals. In cyberspace ID systems the sensors are different because the environmental dimension is different. Instead of a missile launch and supersonic transport through the atmosphere, cyberspace sensors observe information flowing in networks. However, just as C2 operational personnel are interested in the origin, velocity, threat, and targets of a warhead, network-security personnel are interested in the identity, rate of attacks, threats, and targets of malicious intruders and criminals[2]. Input into next-generation IDSes consists of sensor data, commands, and a priori data from established databases. For example, the system input would be data from numerous distributed packet sniffers, system log files, SNMP traps and queries, signature-based ID systems, user-profile databases, system messages, threat databases, and operator commands. (See Figure 4.)

The output of fusion-based ID systems consists of estimates of the identity (and possibly the location) of a threat source, the malicious activity, taxonomy of the threats, the attack rates, and an assessment of the potential severity of damage to the projected target(s). We extrapolated from Waltz[12] to suggest possible generic sensor characteristics of next-generation network fusion systems[2]:

- Detection Performance is the detection characteristics – false-alarm rate, detection probabilities, and ranges – for an intrusion characteristic against a given network-centric noise background. For example, when detecting malicious activity, nonmalicious activity is typically modelled as noise.
- Spatial/Temporal Resolution is the ability to distinguish between two or more network-centric objects in space or time.
- Spatial Coverage is the span of coverage, or field of view, of the sensor (i.e., the spatial coverage of a system log file is the computer system processes and system calls being monitored).
- Detection/Tracking Mode is the mode of operation of the sensor (i.e., scanning, single or multiple network object tracking).
- Target Revisit Rate is the rate at which a network object or event is revisited by the sensor to perform measurements.
- Measurement Accuracy is the statistical probability that the sensor measurement or observation is accurate and reliable.
- Measurement Dimensionality is the number of measurement variables for network object categories.
- Hard vs. Soft Data Reporting is the decision status of the sensor reports. (I.e., can a command decision be made without correlation, or does the sensor require confirmation?)
- Detection/Tracking Reporting is the characteristic of the sensor with regard to reporting events. (Does the sensor maintain a time-sequence of the events? type of historical event buffers?)

In our fusion model, situational data is collected from network sensors with elementary observation primitives; identifiers, times of observation, and descriptions. The raw data requires calibration and filtering, referred to as Data Refinement (short-term knowledge). Object Refinement is a process that correlates data in time (and space if required); the data is assigned appropriate weighted metrics. Observations may be associated, paired, and classified according to intrusion-detection primitives.

Situation Refinement (mid-term knowledge) provides situational knowledge and awareness after objects have been aligned, correlated, and placed in context in an object base. Aggregated sets of objects are detected by their coordinated behavior, dependencies, common points of origin, common protocols, common targets, correlated attack rates, or other high-level attributes.

In the interdomain construct of Figure 1, network objects and data flows will be identified and tracked by placing sensors at or between the interdomain gateways. Without going into the details, it can be shown that temporal resolution of the cyberspace situational awareness is directly proportional to the ratio of the transit time of the datagram and the sensory fusion process and inference time.

As an analogy we offer the tracking of an object in aerospace – for example, a projectile. If the intercept time of a projectile is greater than the time used by radar or another tracking system and other required processing, then it is not possible to track and react to the object before the projectile hits the target. For example, if the datagram will reach its destination in 30ms, then the decision-fusion process required for network situational awareness must be much less than 30ms. Highly critical situational awareness can be achieved by networking the sensors (and optional command and control links) out-of-band. Current-generation systems use in-band processing, which can only achieve limited temporal resolution.

Extensible Threat Taxonomy Fusion

The number of IP packets processed by the Internet gateways of Figure 5 is enormous. Gateway sensors acquire and forward proportionally large amounts of data to packet analysis and correlation processes. For example, a router processing 100,000 packets per second on a high-speed interface, logging 14 bytes of information per packet, produces approximately 1.4 MBPS of data per sensor. It is clear that distributed sensors in network-centric IP fusion systems require local processing. Consequently, sensor output data should be reduced at the sensor to minimize central fusion processing and transport overhead costs.

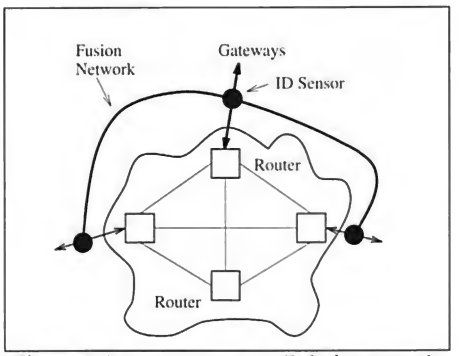


Figure 5. Gateway sensors on ID fusion network

We focus here on the sensor output by outlining an example extensible taxonomy framework of TCP/IP-based threats. Antony[14] discusses database requirements for fusion system and situational knowledge. He states that knowledge is either declarative or procedural. Declarative knowledge is passive factual knowledge or knowledge of relationships (e.g., files). Procedural knowledge is a special case of declarative knowledge represented as patterns, algorithms, and transformations.

Entity relationships are the most fundamental declarative models for sensor data representation. Binaries trees, family trees, and general taxonomies are examples of the elemental database relationships required for situational analysis; the vast majority can be represented by the SQL command[14]:

```
SELECT(attribute) FROM (table) WHERE (condition)
```

With this basic database model and data-selection primitives in mind, we offered a framework TCP/IP threat taxonomy[3]. This framework was offered as an extensible

context-dependent TCP/IP threat tree based on the SNMP management information base (MIB) concept. The SNMP MIB concept for representing context-dependent data is well suited for network-centric threats (and countermeasures).

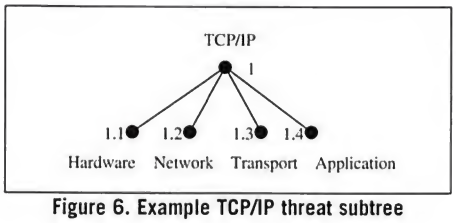


Figure 6. Example TCP/IP threat subtree

Threats to TCP/IP at the physical layer are service disruptions caused by natural disasters such as fires or flooding, cuts to cables, malfunctioning transceivers, and other hardware failures. Threats to the network layer include IP source-address spoofing and route-cache poisoning. An extensible context-dependent framework for this is illustrated in Figures 6, 7, and 8.

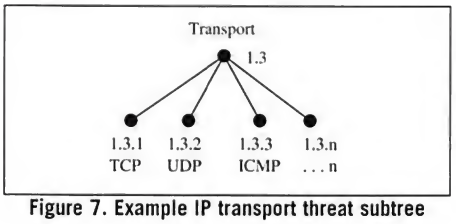


Figure 7. Example IP transport threat subtree

Three primary data flows (services) exist on the Internet: User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Internet Control Message Protocol (ICMP)[1]. Domain Name System (DNS) cache poisoning and UDP port-flooding denial-of-service attacks are examples of two vulnerabilities exploited using UDP services. The ping-of-death and ICMP redirect bombs are examples of Internet attacks based on ICMP. TCP vulnerabilities include TCP sequence number and SYN flood attacks, as illustrated in Figure 8.

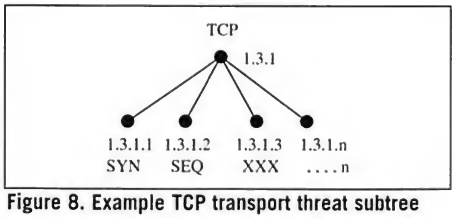


Figure 8. Example TCP transport threat subtree

Security threats and countermeasures can be represented using the ASN.1 MIB notation. For example, a TCP SYN flood attack could be represented with the following OBJECT IDENTIFIER (OID):

```
tcpSYNFlood OID ::= { iso 3.6.1.5.1.3.1.1 }
```

Additional sub-object examples for `tcpSYNFlood OID` could be the source address or the target address of the malicious SYN packet and a counter with the number of SYN floods:

```
tcpSYNFlood.source OID ::= { iso 3.6.1.5.1.3.1.1.1 }
tcpSYNFlood.dest OID ::= { iso 3.6.1.5.1.3.1.1.2 }
tcpSYNFlood.number OID ::= { iso 3.6.1.5.1.3.1.1.3 }
```

Developing an extensible TCP/IP security threat MIB is a solid first step on the road to creating Internet ID fusion systems. Other long-term knowledge databases include context-dependent countermeasure, threat profiles, and attack-capabilities databases.

Conclusion

Future reliable services that provide long-term threat, countermeasure, and other security-related information to fusion systems are similar to the current state of the art of weather forecasting and threat assessment. Fusion from multiple short-term sensors further processed with long-term knowledge creates short mid-term situational awareness. Situational awareness is required to operate and survive in a complex world with both friendly and hostile activities.

All intelligent biological organisms fuse short-term and long-term knowledge to create situational awareness. Humans continually create and redefine systems that help us increase and refine our situational knowledge. These systems include air traffic control, battlefield management, early-warning systems, and robotics. There are strong indications, based on our work in both the Air Force and commercial industry, that future ID systems will shift toward more advanced fusion-based models.

Our crystal ball is as foggy as yours, but if the developments in situational awareness systems in air traffic control over the past 40 years are any indication, then Internet traffic-control systems and next-generation intrusion-detection systems have a significant and challenging future in store for all of us.

References

[1] Stevens, R. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 1994.

[2] Bass, T. "Intrusion Detection Systems and Multisensor Data Fusion: Creating Cyberspace Situational Awareness." *Communications of the ACM.* Forthcoming, 1999.

[3] Bass, T. "Multisensor Data Fusion for Next Generation Distributed Intrusion Detection Systems." 1999 IRIS National Symposium on Sensor and Data Fusion, May 1999.

[4] Bass, T.; Freyre, A.; Gruber, D.; and Watt., G. "E-Mail Bombs and Countermeasures: Cyber Attacks on Availability and Brand Integrity." *IEEE Network*, March/April 1998, pp. 10-17.

[5] Denning, D. "An Intrusion-Detection Model." *IEEE Transactions on Software Engineering*, February 1987, pp. 222-232.

[6] Mukherjee, B.; Heberlein, L.; and Levitt, K. "Network Intrusion Detection." *IEEE Network Magazine*, May/June 1994, pp. 26-41.

[7] Denning, D., et al. "A Prototype IDES: A Real Time Intrusion Detection Expert System." Computer Science Laboratory, SRI International, August 1987.

[8] Snapp. S. et al. "A System for Distributed Intrusion Detection." *Proceedings of IEEE COMPCON*, March 1991, pp. 170-176.

[9] Bauer, D. and Koblentz, M. "NDIX – An Expert System for Real-Time Network Intrusion Detection." *Proceedings of the IEEE Computer Networking Symposium*, April 1988, pp. 98-106.

[10] Hochberg et al. "NADIR: An Automated System for Detecting Network Intrusion and Misuse." *Computers & Security*, Elsevier Science Publishers, 1993, pp. 235-248.

[11] Heberlein, L. et al. "A Network Security Monitor." *Proceedings of the IEEE Computer Society Symposium*, Research in Security and Privacy, May 1990, pp. 296-303.

[12] Waltz, E., and Llinas, J. *Multisensor Data Fusion.* Boston: Artech House, 1990.

[13] Waltz, E. *Information Warfare Principles and Operations.* Boston: Artech House, 1998.

[14] Antony, R. *Principles of Data Fusion Automation.* Boston: Artech House, 1995.

on reliability

Security and Reliability

by John Sellens

John Sellens is part of the GNAC Canada team in Toronto, and proud to be husband to one and father to two.

<jsellens@gnac.com>

The article published under this title in the August issue (Vol. 24, No. 4) was NOT what it was advertised to be, but another article inserted by error. Here is John Sellens's actual article, with our apologies for any incovenience and confusion we have caused through this clear instance of unreliability.

I will attempt to provide an overview of how "security" contributes to the reliability of your systems (and, I hope, show how a lack of security decreases reliability). Some of the topics covered relate to discussions in the previous articles in this series; this is an attempt to gather everything related to security and reliability in one place, which should (in theory) make it easier to see the big picture.

Security is a wide-ranging and sometimes poorly defined topic. "Computer people" often (incorrectly) think of security as being related only to things that you can do with a keyboard, a computer, random bits and bytes, and someone else's password. Accordingly, I will attempt to summarize what security is, or at least what it is in the context of this article. The relevant security-related elements I will cover are:

Access control – passwords and other mechanisms that attempt to require some level of authentication and authorization for access to your networks and systems (i.e., how to know when to open the barn door).

Physical security – protection against physical attacks and "acts of God."

Intrusion detection – how to detect when someone unexpected has entered through an open or insufficiently closed barn door.

Correction – fixing things when they break, which includes "remotivating" individuals when they act inconsistently with what is expected.

Change management – to ensure that changes are appropriate and have been subjected to the appropriate review and approval.

Security is not just "prevention" – it's prevention, detection, control, and correction. And when you have all those, you have (of course) a more reliable system. Let's review each of the five elements in turn.

Access Control – Authentication and Authorization

Please allow me to be ridiculous for a moment: If you have no access control, anyone can do anything to your systems, and so they are almost by definition unreliable. And you're similarly exposed even if you have good access control but have no authorization mechanism that limits what different users can do.

I'm going to discuss access control in two parts, authentication and authorization. I'll further subdivide the discussion of authorization into logical access restrictions, physical access restrictions, and activity restrictions.

Authentication

Authentication is what we are all (I hope) familiar with – some form of userid and password pair that "proves" that the user is who he or she claims to be. In theory, both the userid and password could change with time, but the most common implementations involve a publicly known userid and a static password. (I'll define a static password as one that stays the same until it is explicitly changed, typically by the user.)

Static passwords are most commonly stored on the destination machine (or network of machines), typically in an encrypted or obscured form to prevent the casual browsing of passwords. The most commonly used "more secure" static-password mechanism is

kerberos, by which the passwords are stored on a "secure" server, and the protocol protects the passwords as they are passed back and forth between various clients and servers to authenticate users. One problem with this is the weakest-link problem – you need to have kerberos locally available on every device, or you risk sending your password over some connection in cleartext, which limits the effectiveness of kerberos in those environments. (You can sometimes secure otherwise cleartext links with ssh encryption, but then you need to have ssh on the local box, which is the same problem but with a different piece of software.)

More advanced (i.e., obfuscated or annoying) systems use some form of one-time-password (OTP) system to guard against password eavesdropping (over the shoulder or over the network) and password sharing. Some OTP systems are software only (such as S/KEY[1]) but the more common approach requires the use of some form of token, which computes or reports the next password in the series; the most commonly used token is probably SecurID from Security Dynamics. These systems have some mechanism to guard against password reuse and typically also require some sort of secret PIN in addition to the token in order to authenticate.

The reason for an authentication mechanism is to identify the user, and the more reliable the authentication mechanism, the more reliable your overall system is going to be, because you have a better "front-door" defense to protect you from the unreliable among us. I'm a big fan of one-time passwords – only the most rudimentary of systems would not benefit from the use of OTPs, even if used only for authenticating the more privileged users or for granting root (or equivalent) access.

I'm a big fan of one-time passwords – only the most rudimentary of systems would not benefit from the use of OTPs.

Authorization

The next element in an access-control system is what I will refer to as "logical access restrictions." Logical access restrictions are those that are based on such things as the originating network address of a connection, time of day, or current usage rules. The most common way to implement originating network restrictions (in the UNIX world at least) is through the use of the "TCP wrapper"[2] package, which makes it easy to "wrap" certain services (such as telnet) with an access-control program that can restrict on the basis of network address, etc. The other logical restrictions are more commonly implemented with certain operating-system configurations, custom shells, or commercial software. Restrictions that you might want or need to implement include:

- No multiple logins – You may wish to limit concurrent access to particular applications or systems for reasons of system load, security, or (business) process control.
- No logins from multiple (apparent) locations – You may wish to prevent users from being in two places at once, primarily for security reasons. If a user is in your office, working away, it might be safe to conclude that a login attempt from halfway around the world was not actually the same person who is authorized to use your system. Of course, a connection from two different locations doesn't automatically mean that it's two different people, since the person could have connected to the remote machine over the network before connecting back, but it would mean your traffic might be taking a long, possibly exposed route, which probably isn't a good thing either.
- No off-hours connections – It's probably not reasonable (or expected) for an ordinary accounts-payable clerk to be doing a check-printing run at 2:00 am Sunday morning; you might want to use "off-hours" restrictions to prevent that before it happens.
- No connections during maintenance periods – Sometimes you need the machine to

Allowing users on during system maintenance can sometimes just make things more complicated.

be up and running but don't want to allow (ordinary) users to sign on. Allowing users on during system maintenance can sometimes just make things more complicated. One classic example is doing full backups to a network backup server in preparation for an OS upgrade. On UNIX, this can sometimes be accomplished with the /etc/nologin file (but often not).

- Peak-load restrictions – You might wish to refuse additional logins if the system load (however that is calculated) is above some threshold. This can help avoid making an already bad situation even worse.

As a more concrete example, I'll mention that I once implemented an access-control system that made use of almost all those restrictions. The system was a job database for university students, serving several thousand students each term, and the usage was very "peaky" – there were a few periods of very high demand and long periods of limited use. We wanted to make sure that each student was using only one session at a time and that we didn't let on too many simultaneous users (because otherwise the system would crawl to a halt); and we wanted to be able to prevent user signons during the daily update window and during periods of "emergency" maintenance. We did all that with, a simple shell script, which we used as each student's login shell, proving that (once again) complicated solutions are not always required.

It's probably worth mentioning that mechanisms and policies like this have a long history in the mainframe world.

Complementary to logical access restrictions are (of course) physical access restrictions. Sometimes you wish to allow access only to a particular system, application, or function if the user is (thought to be) in a "secure" location. For UNIX systems the most common example of this kind of restriction is to allow direct root logins only from the system console. Other examples include allowing connections only from within your building, enforced either through the use of hardwired connections (almost unheard of in these days of networked workstations), subnets and firewalls, or simply not allowing any connections (network or dialup) to and from the outside.

The final component of access control that I am going to cover are what I'll call "activity restrictions" – restrictions or limits on the commands and functions that a user can invoke. These come into effect once your authentication system has identified a particular user, and the user (or the user's connection) has passed any logical or physical access restrictions that have been implemented.

One of the most common (UNIX) examples of activity restrictions is the common requirement that a user be a member of a certain group (often "wheel" or group 0) in order to "su" to root. Lots of other examples of group- or ACL-based restrictions exist. Other restrictions can be implemented by applications, using compiled in information (bad), or files or database entries with restriction or permission information.

I suggest dividing activity restrictions into three types:

- Static – yes/no restrictions, independent of other considerations, such as date or time, other users, etc. Some examples are: group membership requirements, as mentioned above; the prevention of access to a general-purpose environment by such mechanisms as menuing systems; restricted shells.
- Variable – restrictions that are based on straightforward but varying information, such as date or time, connection origin, etc. Some examples are: no recreational Web sites during business hours; not allowing check printing outside regular hours; no root access if you're not on the local network.

- Complex – restrictions that are controlled by other events, situations, status, etc. Some examples are: permission granted or denied on the basis of file or database contents; task allowed only at certain steps in a business process; there must be an operator on duty before the operation is allowed.

You will have noticed that the line between activity restrictions and logical and physical access restrictions gets a little blurry sometimes.

I don't claim to have covered all situations here. One obvious situation that's not covered is multiple authentication, where two or more people must agree and authenticate before a task is executed. (Recall those action movies featuring nuclear missile silos where two people have to turn two different keys on opposite sides of the room at the same time, and they're both carrying guns.) And I haven't mentioned the use of biometrics for authentication.

Some of these access-control mechanisms can be quite inconvenient and/or obtrusive. As in most other discussions of reliability, there's a tradeoff between reliability and control on the one hand and cost and inconvenience on the other, and each organization must strike the most appropriate balance for its needs. And I haven't mentioned the need for proper logging, which is a necessity for tracking, troubleshooting, and change control.

And to tie this discussion back to reliability, good access control means that you limit, control, or track who did (or could do) what, when, and under what circumstances. This means that when you determine that certain controls or limits are required to help your systems, networks, and business processes function reliably, you've got (at least part of) the mechanism to help you implement them.

In most cases, physical access to a machine is tantamount to administrator access.

Physical Security

The preceding discussion has focused primarily on electronic access to systems and networks, which is the traditional area of concern for computer-oriented people. But it's just as important to consider the physical security aspects, and again balance the costs (monetary and otherwise) against the expected risks and/or advantages. Note that I'm not talking here about disaster-recovery planning, or high-availability hardware – I'm talking about preventing people (or things) from getting physical access to your premises or equipment.

Why is physical security important? In most cases, physical access to a machine is tantamount to administrator access. In the most extreme cases, a machine (or parts of it) is stolen and attacked at the thief's leisure, whim, or screwdriver. Physical security can also help to guard against so-called acts of God – a more secure building is likely to be stronger and more appropriately located.

What kinds of things should physical security guard against, and how do they contribute to reliability?

- Equipment theft – If your computers, disks, or network hubs are missing, it's hard to offer a reliable service. It's also getting fairly common for people to steal internal components, such as processors, memory, or disks, as they're easier to carry, and often not immediately obviously missing.
- Media theft – Removable media (disks, tapes, cartridges, etc.) can contain very useful information (consider the backups of your customer database) and are often easy to carry and not likely to be noticed missing very quickly. The business risks here are obvious, but the impact on reliability is less so. Reliability can be reduced by the pos-

Quite simply, if you can't detect when something has gone awry, you've got much less chance of protecting yourself and your systems.

sibility of the use of confidential information to attack the organization at a later date and by the lack of backups, which could make recovery a very painful process.

- Console or network access – Unauthorized access to console ports or network connections or devices can open you up to all sorts of attacks (such as password sniffing and bug exploitation) that can cause your systems to start behaving unreliably, unintentionally or otherwise.
- Physical destruction – A fire ax can render even the best systems and equipment somewhat unreliable.

What sort of controls should be considered?

- Locks – of varying and appropriate levels of sophistication. It's not unusual to start making the locks more difficult and complicated the closer one gets to the important stuff. Consider the use of ordinary keys that can be copied at the local all-night convenience store, high-security keys that require special blanks and machines to duplicate them, magnetic or other types of access-control cards, combination locks, biometric scans, etc. And remember the hardware that the lock cylinders are attached to – a high-security key cylinder on a $2 latch might not be the most prudent way of securing your machine room.
- Access controls – Electronic logs of who went where and when are handy after the fact and can act as a convenient deterrent; time-of-day restrictions can be used to prevent (or limit) physical access in the middle of the night when no one else is around; and requiring the cooperation of two people to open a particular door all have their place in an access control plan.
- Structural – Check your walls, ceilings, and floors for ease of access and/or destruction. There's no point in putting a high-security lock on a low-security door and frame.
- Monitoring – Consider the use of fire and burglar alarms, video monitoring, etc., but make sure that it's hard for an intruder to get at the video tapes and destroy them and the evidence that they contain.
- More extreme – Some organizations will find it worthwhile to go to greater lengths to secure their premises. Armed guards, "man traps" (small rooms with two independently locking doors that you must pass through when entering or leaving), guard dogs, etc.

Intrusion Detection

The best security system in the world is reduced in its effectiveness if it's not properly monitored. You must have some mechanisms and processes that are designed to detect any intrusions that do take place and, optimally, any attempted intrusions that were blocked by the systems. Proper intrusion-detection systems will alert you when you're under attack and will give you time to increase your awareness or monitoring to fend off any further attacks. For example, if you can detect when a copy of your encrypted passwords have been stolen, you have a better chance of changing all the passwords before they get cracked and exploited and of blocking the access used to intrude. Quite simply, if you can't detect when something has gone awry, you've got much less chance of protecting yourself and your systems. And if you can't protect the systems, it's going to be harder to keep them working reliably.

Techniques and mechanisms for intrusion detection include:

- Log review – Review your log files, either manually or using some (well-protected)

automatic tools, so that you'll have a better chance of noticing the unexpected when it happens.

- Realtime alarm monitoring – Page yourself or the security company when alarms get triggered, or have someone on duty onsite. A quick response to an alarm or alert can limit the amount of damage that happens and can also serve as a deterrent if the attacker is simply looking for a fertile playground, not targeting you specifically.
- Periodic analysis – Tools such as tripwire[3] can notice when files change unexpectedly.

A quick response to an alarm or alert can limit the amount of damage that happens.

Correction

Once you've detected an intrusion or attack (or attempted attack), you need a mechanism and process by which you can put things right again, and, optimally, a way to prevent it from happening again. Keep good backups, know where your distribution media is, have documented procedures and mechanisms to get in touch with the necessary people. Keep up to date on vendor updates, notices, and security alerts in the community at large. Be ready to disconnect machines or networks that are under attack or need repair while you investigate and undertake repairs.

The impact on reliability should be clear – a modified machine or system is at risk, and the sooner you can get things back together, the sooner normal, reliable operation can resume.

The other side of correction is "remotivating" individuals who are acting contrary to policy and reasonable standards of behavior. A user or system administrator who behaves incorrectly (let's say by choosing trivial passwords and writing them on notepaper stuck to their monitor) can be putting other users, systems, and information at risk. If you're expecting people to act appropriately, you had better define and publish what the standards of behavior are and be prepared to enforce or explain them.

Change Management

The best-laid plans can be all for naught if there are no controls around them, and one of the most important controls is change management. The primary components of a proper change-management system are:

- Review – Proper review and testing of any proposed changes will greatly increase the likelihood of a successful, nondisruptive change, and will help prevent intentionally or unintentionally malicious changes from being undertaken. Note that a proper review also ensures that there is proper documentation.
- Authorization – Ensures that changes (to, say, the payroll system) have been properly authorized according to the policies of the organization. Some systems might require only a very low level of authorization to change, but you might want to make sure that any changes to the CEO's laptop are made by a limited set of people.
- Proper implementation – A standard and documented implementation process will help avoid mistakes, will keep downtime to a minimum, and will make your maintenance windows much more bearable.
- Ability to back out – This is often overlooked, but a proper change-management system is prepared to deal with changes that fail (in whatever way) once they are put into production, and ensures that there is a way to get back to the pre-change, working, reliable system.

In Summary

Security is a wide-ranging topic and has an impact on many areas of an organization's activities. Proper security systems, mechanisms, policies, and practices sometimes augment reliability, but in many ways their primary reliability benefit is in preventing the intentional or unintentional reduction of current reliability levels.

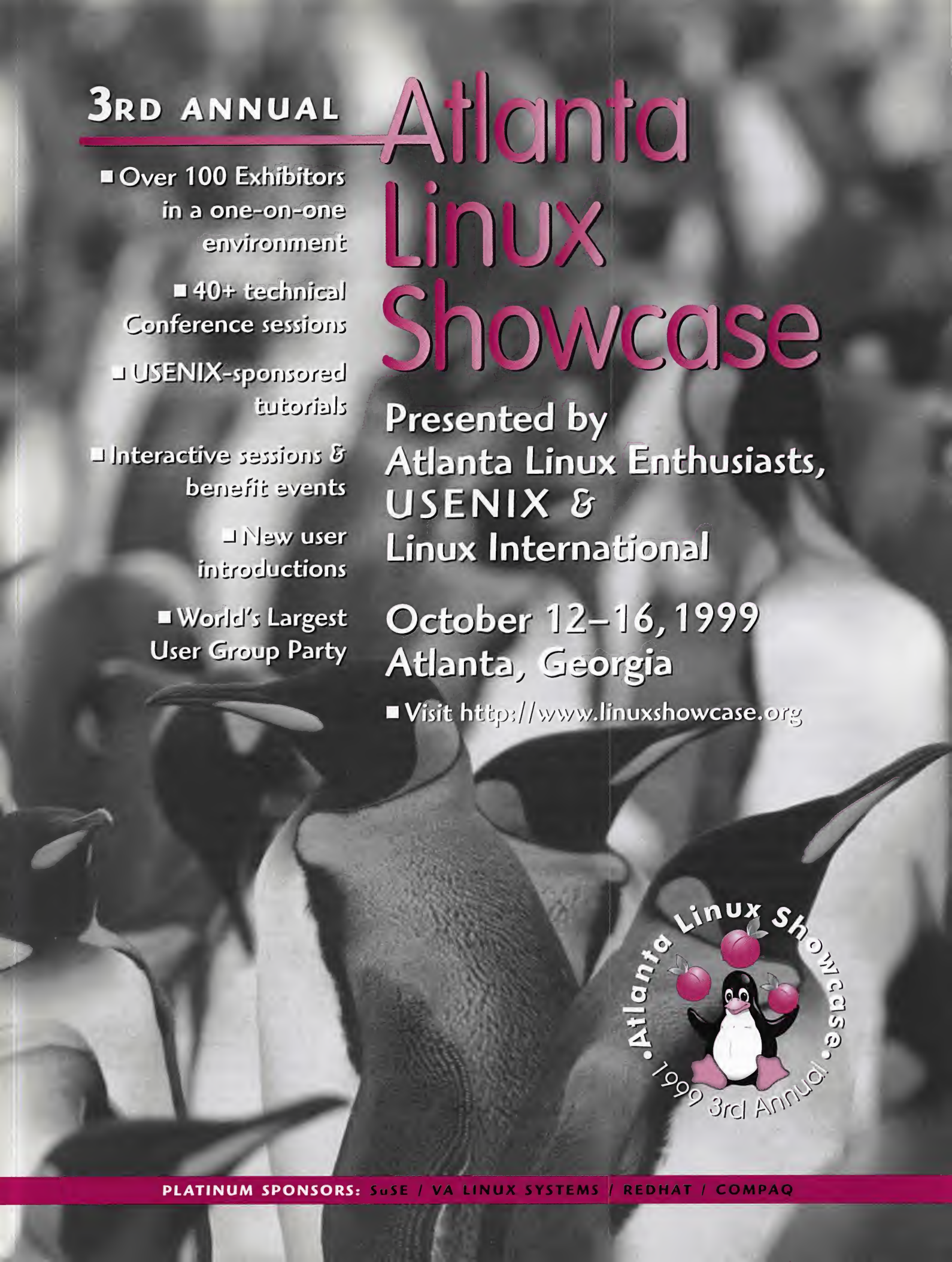
This is the ninth article in the On Reliability series published in *;login:* over the past two years and concludes the list of topics that I had planned to cover. Thanks very much for reading, and I hope you've found this series useful.

References

[1] The S/KEY One-Time Password System from Bellcore. <ftp://ftp.bellcore.com/pub/nmh/>

[2] TCP Wrapper by Wietse Venema (<wietse@porcupine.org>). <ftp://ftp.porcupine.org/pub/security/index.html>

[3] Tripwire, originally by Gene Kim and Gene Spafford, is available from <ftp://info.cert.org/pub/tools/tripwire/> and provides tools to track when system files change unexpectedly.


3RD ANNUAL
Atlanta Linux Showcase
■ Over 100 Exhibitors in a one-on-one environment
■ 40+ technical Conference sessions
■ USENIX-sponsored tutorials
■ Interactive sessions & benefit events
■ New user introductions
■ World's Largest User Group Party
Presented by Atlanta Linux Enthusiasts, USENIX & Linux International
October 12–16, 1999
Atlanta, Georgia
■ Visit http://www.linuxshowcase.org
Atlanta Linux Showcase
1999 3rd Annual
PLATINUM SPONSORS: SuSE / VA LINUX SYSTEMS / REDHAT / COMPAQ

Announcement and Call for Papers **USENIX**

3rd Large Installation System Administration of Windows NT / 2000 Conference

Sponsored by USENIX, the Advanced Computing Systems Association, and by SAGE, the System Administrators Guild

http://www.usenix.org/events/lisa-nt2000

July 30 - August 2, 2000 **Seattle, Washington, USA**

Important Dates

Submission proposals due: *Feb. 16, 2000*
Notification to authors: *March 13, 2000*
Final papers due: *June 1, 2000*

Conference Organizers

Conference Co-Chairs

Phil Cox, *SystemExperts*
Jesper M. Johansson, *Boston University*

Program Committee

Kip A. Boyle, *SRI Consulting*
Alan Epps, *Intel*
Aeleen Frisch, *Exponential Consulting*
John Holmwood, *NOVA Gas Company*
David LeBlanc, *Microsoft Corp.*
Todd Needham, *Microsoft Corp.*
Jason Reed, *System Experts*
Dave Roth, *Microsoft Corp.*
Martin Sjoelin, *Warburg Dillon Read*

Overview

By late next summer, we will have all undergone the flurry or flutter of Y2K, and many of us will be in the midst of another similar dilemma: migrating to Windows 2000. Sites around the world are all slowly beginning to plan this migration, some sooner than others, and all are seeking answers from those who have blazed the trail ahead of them. The Large Installation System Administration of Windows NT conference, LISA-NT, is a forum to bring system administration professionals together to discuss workable solutions to the issues of administering and scaling all versions of the NT environment. In 2000 that will mean migration issues for many of us, and just keeping our heads above water with Windows NT 4.0 for others.

LISA-NT 2000 will bring together peers and experts in our field to discuss leading edge solutions that have a proven track record of working. LISA-NT is put together by and for Windows NT administrators who need solutions to problems such as integration, migration, security, and management using today's technology. We invite you to submit technical papers as well as proposals for invited talks, panel sessions, tutorials, and work-in-progress reports. There are also opportunities for Birds-of-a-Feather sessions and demonstrations of products and solutions. Please review this call for papers, prepare a submission, and join us in making LISA-NT 2000 the premiere conference for system administrators of distributed NT-based environments.

This conference will last four days. Two days of tutorials will be followed by two days of technical sessions including refereed papers, invited talks, and works-in-progress. On August 3-4, 2000, the Fourth USENIX Windows Operating Systems Symposium will be held in the same location.

Topics

LISA-NT is about creating a community of system administration professionals, which come from diverse computing environments, to give them an opportunity to discuss problems and share solutions. Submissions may address administration methodologies (i.e. "This is what I do and why") as well as implemented solutions (e.g. "This is the tool I wrote in Perl or VBscript or any other language" or "This is how I turned my Master-Domain model into a functional Active Directory"). To facilitate these discussions, we encourage you to consider these topics:

Management & Migration

- Management of homogeneous Windows NT networks
- Management of NT in heterogeneous environments
- Large-scale or distributed NT management solutions and experiences
- Locally developed administration tools and techniques
- Models for centralized, remote or distributed management
- Reduction of the total cost of ownership for users and/or workstations
- Backup and restore methodologies
- User account management
- Migrating existing domain structures to Windows 2000 Active Directory
- Large-scale configuration, roll-out, and maintenance of NT workstation and application software
- Management of NT for heterogeneous user constituencies
- Maintaining or enhancing NT security posture during migration to Windows 2000

Sharing and Integration

- Integration of NT into complex, existing environments
- Integration of NT (workstation and/or server) into existing UNIX or Netware environments
- Integration of Unix into NT environments
- Integration of an existing Unix Kerberos Realm with Windows 2000 Active Directory
- Integration of Windows 2000 and Windows NT 4.0 security functionality
- Use of cross-platform distributed services
- Remote access
- Laptops and PDA's running and/or accessing NT
- Deployment of Windows 2000 and/or Windows NT 4.0 in extranets
- Use of IPsec and other VPN solutions in homogeneous or heterogeneous NT/Win2K environments
- Strategies for deployment of Windows 2000 into heterogeneous networks

Tools, Experiences, and Issues

- Successful use of third-party applications during daily administration
- Surveys of examined tools (good and bad) while implementing a solution

- Performance tuning and measurement
- E-mail management
- Central name service and browsing
- Software licensing, deployment, and upgrade management
- Security issues, including policies, education, response, and fixes
- The registry

Best Paper Awards

Awards will be given for the best paper and best student paper at the conference.

What to Submit

The program committee seeks submissions from the Windows NT system administration community in the following formats:

Technical White Papers

We seek papers relating work of general interest to system administrators of Windows NT, particularly technical papers that reflect hands-on experience or describe implemented or attainable solutions. Submissions will be judged on the quality of the written submission and whether or not the work advances the art and science of NT system administration.

A paper submission should:

- Contain a short abstract
- Include an outline of the paper. If you have a completed paper, you may submit it instead of the abstract and outline.
- Conform to the "How and Where..." instructions below

Please see the detailed author instructions, which include a sample abstract, for more information: *http://www.usenix.org/events/lisa-nt2000/cfp/guidelines.html*

Authors of an accepted paper must provide a final paper for publication in the conference proceedings. At least one author of each accepted paper will present the paper during the technical track of the conference. These presentations generally include a 20-minute talk with 5-10 minutes of questions from the audience.

Conference proceedings containing all full technical white papers will be distributed to attendees and, following the conference, will be available online to USENIX members and for purchase.

The LISA-NT Conference, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Invited Talks/Panel Sessions

These survey-style talks given by experts range over many interesting and timely topics. Invited talks fit in a 90-minute session with a short question and answer session at the end. Please submit a proposal for an invited talk or a panel session to: *lisantpapers@usenix.org*

Work-in-Progress Reports (WIPs)

Work-in-Progress Reports (WIPs) are short talks that introduce new or ongoing work. If you have work you would like to share or an interesting idea that is not quite ready to be published as a paper, a WIP is for you. Acceptance will be based upon the applicability and scalability of the proposed solution. To submit a WIP, email a description of the problem and your (possibly incomplete) solution if necessary, and why the problem you are addressing is interesting, to: *lisantpapers@usenix.org*

Tutorials

On July 30 and 31, there will be full and half-day tutorials in all areas and levels of expertise for Windows NT system administrators. Previous tutorial sessions have covered topics such as "Windows NT Security," "Windows NT Internals," "Configuring Samba, Avoiding Common Pitfalls," and "Administering Windows NT DHCP and DNS servers."

If you are interested in presenting a tutorial at LISA-NT, please contact the USENIX tutorial coordinator:

Daniel V. Klein
Email : *dvk@usenix.org*
Phone : +1.412.422.0285
Fax : +1.412.421.2332

How and Where to Send Submissions

Please email your submission to *lisantpapers@usenix.org* in one of the following formats (listed in order of preference). If you enclose files as an attachment to your submission, please use MIME encoding.

- RTF
- Microsoft Word 97
- Postscript formatted for 8.5" x 11" page
- HTML
- Plain text with no extra markup

A cover letter with the following required information must be included with all submissions:

Authors : Names and affiliation of all authors, and an indication of which, if any, are full-time students
Contact: Primary contact for the submission
Address: Contact's full postal address
Phone: Contact's telephone number
Fax: Contact's fax number
Email: Contact's e-mail address
URL: For all speakers/authors (if available)
Category: Category of the submission (paper, invited talk, panel, WIP, BoF, demo/poster session)
Title: Title of the submission
Needs: Audio-visual requirements for presentation

We will acknowledge receipt of a submission by email within one week.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in May 2000. If you wish to receive the registration materials, please visit the symposium Web site or contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

Announcement and Call for Papers **USENIX**

9th USENIX Security Symposium

Sponsored by USENIX in cooperation with The CERT Coordination Center

http://www.usenix.org/events/sec2000

August 14-17, 2000 **Denver, Colorado, USA**

Important Dates for Refereed Papers

Paper submissions due: *February 10, 2000*
Author notification: *March 23, 2000*
Camera-ready final papers due: *June 15, 2000*

Symposium Organizers

Program Co-Chairs
Steven Bellovin, *AT&T Labs—Research*
Greg Rose, *QUALCOMM Australia*
Program Committee
Carl Ellison, *Intel Corporation*
Ian Goldberg, *UC Berkeley*
Peter Gutmann, *University of Auckland*
Trent Jaeger, *IBM T.J. Watson Research Center*
Markus Kuhn, *University of Cambridge*
Marcus Leech, *Nortel*
Alain Mayer, *Lucent Technologies, Bell Laboratories*
Avi Rubin, *AT&T Labs—Research*
Jeff Schiller, *MIT*
Jonathan Trostle, *Cisco*
Wietse Venema, *IBM T.J. Watson Research Center*
Dan Wallach, *Rice University*
Tara Whalen, *Communications Research Centre Canada*
Elizabeth Zwicky
Invited Talks Coordinator
Win Treese, *Open Market Inc.*

Symposium Overview

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in security and applications of cryptography.

Dr. Blaine Burnham, director of the Georgia Tech Information Security Center (GTISC), will give the keynote address. Dr. Burnham most recently served as program manager for the National Security Agency (NSA) at Ft. Meade, Maryland.

If you are working in any practical aspects of security or applications of cryptography, the program committee would like to urge you to submit a paper. Submissions are due on February 10, 2000.

This symposium will last four days. Two days of tutorials will be followed by two days of technical sessions including refereed papers, invited talks, works-in-progress, panel discussions, and a two-day exhibition.

Symposium Topics

Refereed paper submissions are being solicited in all areas relating to system and network security, including but not limited to:

- Adaptive security and system management
- Analysis of malicious code
- Applications of cryptographic techniques
- Attacks against networks and machines
- Authentication and authorization of users, systems, and applications
- File and filesystem security
- Firewall technologies
- Intrusion detection
- IPSec and IPv6 security
- Public key infrastructure
- Rights management and copyright protection
- Security in heterogeneous environments
- Security incident investigation and response
- Security of agents and mobile code
- Techniques for developing secure systems
- Trust management
- World Wide Web security

Papers covering "holistic security"—systems security, the security of entire large application systems, spread across many subsystems and computers, and involving people and environment—are particularly relevant. On the other hand, papers regarding new cryptographic algorithms or protocols, or electronic commerce primitives, are encouraged to seek alternative conferences.

Refereed Papers

Papers that have been formally reviewed and accepted will be presented during the symposium and published in the symposium proceedings. The proceedings are provided free to technical session attendees. Additional copies will be available for purchase from USENIX.

Best Paper Awards

Awards will be given at the conference for the best paper and for the best paper that is primarily the work of a student.

Tutorials, Invited Talks, WIPs, and BoFs

In addition to the refereed papers and the keynote presentation, the technical program will include tutorials, invited talks, panel discussions, a Work-in-Progress session (WIPs), and Birds of a Feather Sessions. You are invited to make suggestions regarding

topics or speakers for any of these formats to the program chair via email to *securitychairs@usenix.org.*

Tutorials

Tutorials for both technical staff and managers will provide immediately useful, practical information on topics such as local and network security precautions, what cryptography can and cannot do, security mechanisms and policies, firewalls and monitoring systems.

If you are interested in proposing a tutorial, or suggesting a topic, contact the USENIX Tutorial Coordinator, Dan Klein, by phone at +1.412.422.0285 or by email to *dvk@usenix.org.*

Submitting an Invited Talk Proposal

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talk Coordinator, Win Treese, welcomes suggestions for topics and requests proposals for particular talks. In your proposal state the main focus, including a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to *securityit@usenix.org.*

Work-in-Progress Session (WIPs)

The last session of the symposium will be a Work-in-Progress session. This session will consist of short presentations about work-in-progress, new results, or timely topics. Speakers should submit a one- or two-paragraph abstract to *securitywips@usenix.org* by 6:00 pm on Wednesday, August 16, 2000. Please include your name, affiliation, and the title of your talk. The accepted abstracts will appear on the conference Web page after the symposium. The time available will be distributed among the presenters with a minimum of 5 minutes and a maximum of 10 minutes. The time limit will be strictly enforced. A schedule of presentations will be posted at the symposium by noon on August 17. Experience has shown that most submissions are usually accepted.

Birds-of-a-Feather Sessions (BoFs)

There will be Birds-of-a-Feather sessions (BoFs) both Tuesday and Wednesday evenings. Birds-of-a-Feather sessions are informal gatherings of persons interested in a particular topic. BoFs often feature a presentation or a demonstration followed by discussion, announcements, and the sharing of strategies.

How and Where to Submit Refereed Papers

Papers should represent novel scientific contributions in computer security with direct relevance to the engineering of secure systems and networks.

Authors must submit a mature paper in PostScript format. Any incomplete sections (there shouldn't be many) should be outlined in enough detail to make it clear that they could be finished easily. Full papers are encouraged, and should be about 8 to 15 typeset pages. Submissions must be received by February 10, 2000.

Along with your paper, please submit a separate email message in ASCII containing:

- The title, all authors of the manuscript, and their affiliations.
- The name of one author who will serve as a contact, with regular and electronic mail addresses, daytime and evening telephone numbers, and a fax number.
- Indicate any authors who are full-time students.

For more details on the submission process, authors are encouraged to consult the detailed author guidelines on the symposium website at: *http://www.usenix.org/events/sec2000/.*

All submissions will be judged on originality, relevance, and correctness. Each accepted submission may be assigned a member of the program committee to act as its shepherd through the preparation of the final paper. The assigned member will act as a conduit for feedback from the committee to the authors. Camera-ready final papers are due on June 15, 2000.

Authors will be notified of acceptance by March 23, 2000.

The Security Symposium, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Specific questions about submissions may be sent to the program chairs via email to *securitychairs@usenix.org.*

For reliability, please send one copy of your paper to the program committee via **both** of the following two methods. All submissions will be acknowledged.

1. **Email (PostScript) to:**
   securitypapers@usenix.org
2. **Send a hard copy to:**
   Security Symposium
   USENIX Association
   2560 Ninth Street, Suite 215
   Berkeley CA 94710
   U.S.A.
   Phone: +1.510.528.8649

Security 2000 Exhibition

Demonstrate your security products to our technically astute attendees responsible for security at their sites. Meet with attendees in this informal setting and demonstrate in detail your security solutions. We invite you to take part. Contact: Dana Geffner, Email: *dana@usenix.org,* Phone: +1.831.457.8649

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in May 2000. If you wish to receive the registration materials, please visit the symposium Web site or contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

send email to jel@usenix.org

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are December 1, 1999, and February 2, 2000.

USENIX The Advanced Computing Systems Association

;login:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710